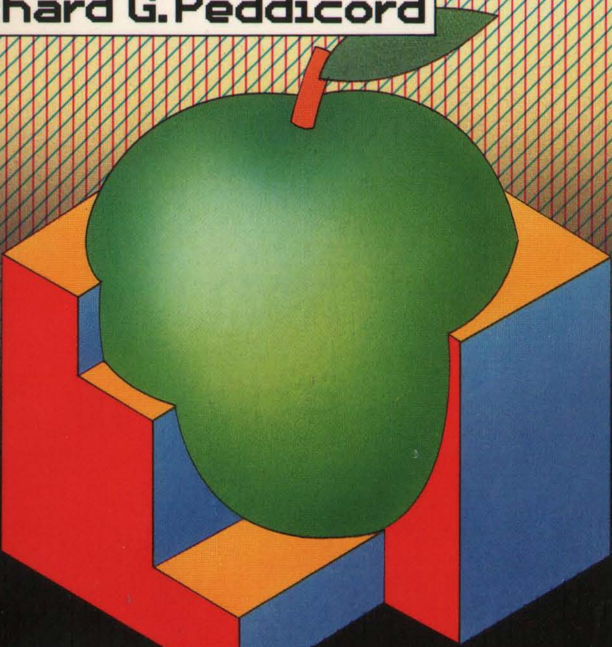
GOLDMANN

computer compact

APPLE BASIC

System und Anwendung

Richard G. Peddicord



computer compact

Richard G. Peddicord

APPLE BASIC

System und Anwendung

Wir danken der Firma Apple Computer für die freundliche Überlassung des Bildmaterials.

Printed in Germany · 12/84 · 1. Auflage · 1110
© 1984 by Alfred Publishing Co. Inc.
© 1984 der deutschen Ausgabe bei Wilhelm Goldmann Verlag, München
Umschlaggestaltung: Design Team München
Übersetzung: Steffen Hölldobler
Konzeption und Redaktion: topic GmbH, München-Karlsfeld
Herstellung: Claus Seitz/Peter Sturm
Satz: Fotosatz Skazel, München
Druck: Presse-Druck Augsburg
Verlagsnummer: 13120

ISBN 3-442-13120-0

INHALT

1. Einführung

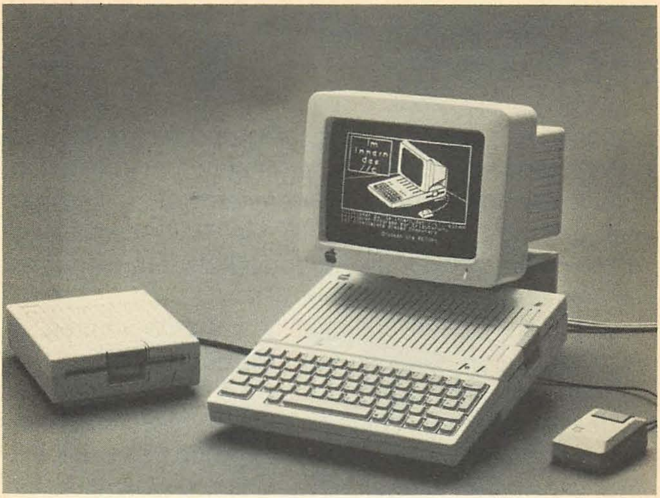
Der Apple II zählt nach wie vor zu den populärsten Mikrocomputern auf dem Markt. Die Gründe für seine Beliebtheit liegen nicht nur in seiner Leistungsfähigkeit und in seinem günstigen Preis, sondern auch in Applesoft BASIC, einer Version der verbreiteten und bekannten Programmiersprache, die auf diesem Computer läuft. Applesoft BASIC besitzt gegenüber anderen BASIC-Versionen zahlreiche Vorteile; es gestattet das Schreiben mehrerer Anweisungen in einer Zeile, die Verwandlung langer Variablennamen oder das Weglassen bestimmter Parameter, weshalb vor allem Anfänger sich ihr gerne zuwenden. Der vorliegende Band aus der Reihe »Goldmann computer compact« bietet eine leicht verständliche und praxisbezogene Einführung in Applesoft BASIC. Er setzt allerdings voraus, daß Sie die angebotenen Beispiele und Übungen auch tatsächlich an einem Apple II durcharbeiten und ausprobieren. Nur dann gewinnen Sie die nötige Vertrautheit mit den Eigenheiten von BASIC, seinen Kommandos und natürlich mit dem Apple II selbst, um den Computer schnell und effizient für Ihre Zwecke einsetzen zu können.

2. Ihr Apple II

Für den Apple II steht auf dem Markt eine große Auswahl an Software- und Hardwareprodukten zur Verfügung. Zu den letztlich unentbehrlichen Zusatz- oder Peripheriegeräten zählen Monitor und Diskettenlaufwerk. Vor allem die Diskette als externes Speichermedium setzt den Betreiber eines Mikrocomputers in die Lage, auch komplizierte Programme mit hohem Speicherbedarf auf seinem Gerät laufen zu lassen.

Der Bildschirm

Welchen Bildschirm oder Monitor Sie für Ihren Apple II wählen, hängt von den Aufgaben ab, die Sie mit dem Computer bewältigen möchten. Während für Textverarbeitung beispielsweise ein schwachauflösender Bildschirm durchaus genügt, werden Sie auf Dauer an ihm wenig Freude haben, wenn Sie anspruchsvolle Graphiken erzeugen oder sich mit raffinierten Spielen unterhalten wollen.
Ihr Apple II erlaubt sowohl hoch- wie schwachauflösende Darstellung. In der hochauflösenden Betriebsart stehen Ihnen 280 Spalten und 196 Zeilen zur Verfügung, wobei Sie für jedes Bildelement oder »Pixel« (Kurzbezeichnung für den Ausdruck »Picture Element«) unter sechs Farben auswählen können, nämlich Schwarz, Weiß, Orange, Rosa, Blau und Grün. Je mehr Pixels ein Monitor aufweist, desto höher ist sein Auflösungsvermögen und damit die Qualität der Darstellung; allerdings steigt mit der Anzahl der Pixels wie der verfügbaren Farben der Speicherbedarf rasch an. Dieser Aspekt ist beim Kauf kommerzieller



Eine typische Systemkonfiguration:
Apple II Computer, Monitor und Diskettenlaufwerk.

Programme ebenso zu beachten wie die Tatsache, daß diese in der Regel in Assembler geschrieben sind, einer Programmiersprache, die der binären Maschinensprache nahesteht, mit der Computer letztlich operieren. Da Sie als Betreiber aber in der benutzerfreundlicheren, aber langsameren Programmiersprache BASIC arbeiten, erreichen Sie die Geschwindigkeit nicht, mit der diese kommerziellen Programme ablaufen können; ein Punkt, der bei Computerspielen durchaus von Bedeutung ist.
Die schwachauflösende Darstellung verfügt über 40 Spalten und 40 Zeilen, wobei jedes Pixel eine von 16 verschiedenen Farben zwischen Schwarz und Weiß annehmen kann, deren Auflistung Sie in Kapitel 20 finden. Der Grund, warum bei geringerem Auflösungsvermögen die Farbauswahl zunimmt, hängt mit der Art und Weise zusammen, wie das Bildschirmsignal erzeugt und das Farbbild gespeichert wird.

Die Inbetriebnahme des Systems

Eine kostensparende und vielseitig einsetzbare Systemkonfiguration besteht aus einem Apple II mit Diskettenlaufwerk, der mittels eines RF-Modulators an ein Fernsehgerät angeschlossen ist. Der RF-Modulator wandelt die Videosignale des Computers in Signale um, die für ein Fernsehgerät akzeptabel sind.
Bevor Sie das System in Betrieb nehmen, überprüfen Sie die Anschlüsse. Neben dem Netzkabel verfügt der Apple II über einen sogenannten RCA-Phonostecker, der mit dem Antennenanschluß Ihres Farbfernsehers verbunden wird, womit Sie ihn als Ausgabegerät für Ihren Computer verwenden können. Wollen Sie Ihr Diskettenlaufwerk an Ihren Apple anschließen, so studieren Sie dazu die Instruktionen, die dem Laufwerk beiliegen.
Der Netzschalter Ihres Apple II befindet sich links auf seiner Rückseite. Betätigen Sie diesen Schalter, leuchtet die Anzeige links neben der Tastatur auf.

Die Tastatur

Um sich mit der Tastatur vertraut zu machen, probieren Sie einfach alle Tasten aus. Sie können dabei keinen Schaden anrichten. Erwartet die Maschine eine bestimmte Eingabe, die Sie nicht eintippen, so erscheint eine Fehlermeldung (»?SYNTAX ERROR«), die Sie mittels der RESET-Taste wieder löschen. Diese Taste werden Sie vermutlich sehr häufig benötigen. Drücken Sie sie, so hören Sie einen Piepton aus dem Lautsprecher des Apple, und der aufleuchtende quadratische Kursor erscheint sofort rechts neben dem Applesoft Promptzeichen. Gleichgültig, mit welchem Programm Sie gerade

arbeiten, es befindet sich immer noch im Speicher, und was auch immer Sie zuvor gerade eingegeben haben, es geht nicht verloren. Mit anderen Worten, Sie löschen keine Daten, wenn Sie die RESET-Taste drücken.
Die SHIFT-Taste ist Ihnen von der Schreibmaschine her bekannt: Wenn Sie diese drücken, während Sie eine Taste anschlagen, wird der jeweilige Buchstabe als Großbuchstabe an die Zentraleinheit, das eigentliche »Gehirn« des Computers, geschickt.
Bei Applesoft geben Sie jeweils ein oder mehrere Zeichen ein und drücken dann die RETURN-Taste. Solange Sie diese nicht bedienen, weiß der Rechner nicht, was Sie eingetippt haben. Deshalb können Sie beliebig viele Änderungen an einer Zeile vornehmen, ehe Sie diese an die Zentraleinheit abschicken. Die mit einem Pfeil versehenen Tasten auf der rechten Seite der Tastatur, die auch Kursortasten heißen, sind äußerst nützlich bei der Vornahme von Korrekturen, da sie den Kursor über den Text wandern lassen, ohne diesen zu verändern. Erst wenn Sie eine andere Taste drücken, wird eine entsprechende Änderung vorgenommen. Sie werden von den Kursortasten sehr häufig Gebrauch machen.
Wollen Sie den Kursor über eine größere Anzahl von Zeichen laufen lassen, drücken Sie die REPT-Taste mit einer Kursortaste zugleich. Die REPT-Taste benötigen Sie auch, wenn Sie ein Zeichen mehrfach schreiben wollen. Einige Kommandos, die Sie unbedingt kennen sollten, sind im folgenden angegeben:

HOME Es löscht den Bildschirm und setzt den Kursor in die linke obere Ecke des Bildschirms.

DEL Setzen Sie nach DEL eine Zeilennummer Ihres Programms, gefolgt von einem Komma sowie einer weiteren Zeilennummer, so werden sämtliche Programmzeilen zwischen den angegebenen Zeilennummern gelöscht.

LIST Es bewirkt die Auflistung Ihres Programms.

RUN Es bewirkt die Ausführung Ihres Programms.

PR#0 Abschalten des Druckers.

PR#1 Ansteuerung des Druckers über Anschluß 1.

PR#6 Ansteuerung des Diskettenlaufwerks über Anschluß 6.

Übungen

Führen Sie folgende Anweisungen aus:

a) Drücken Sie dreimal die RESET-Taste.
b) Drücken Sie zweimal die Leertaste.
c) Drücken Sie die RETURN-Taste einmal.
d) Drücken Sie zehnmal den Buchstaben M.
e) Drücken Sie die linke Kursortaste zehnmal.
f) Tippen sie PRINT »MITTWOCH«.
g) Drücken Sie zehnmal die RETURN-Taste.

3. Ein druckender Tischrechner

Applesoft BASIC bietet eine Betriebsart, die sich als sehr nützlich für die Durchführung von Berechnungen erweist. Setzen Sie vor einen für den Rechner verständlichen arithmetischen Ausdruck das Kommando PRINT, so gibt er nach Betätigung der RETURN-Taste das Ergebnis dieses Ausdrucks aus.

Addition und Subtraktion

Um diese Fähigkeit Ihres Computers zu überprüfen, versuchen Sie folgende Eingabe:

```
PRINT 5+6
```

Bedienen Sie jetzt die RETURN-Taste, so erhalten Sie das Ergebnis:

```
11
```

Man kann auch mehrere Additionen und Subtraktionen in einem Ausdruck zusammenfassen. Verwenden Sie dabei keine Klammern, so arbeitet die Maschine den Ausdruck von links nach rechts ab. Um

```
PRINT 5+6-7-3+4
```

zu berechnen, addiert die Maschine 5 und 6, vom Zwischenergebnis 11 zieht sie zunächst 7, dann 3 ab und addiert schließlich 4 hinzu. Sie erhalten als Ergebnis die Zahl 5. Durch die Verwendung von Klammern läßt sich die

Bedeutung des Ausdrucks ändern, beispielsweise in folgender Weise:

```
PRINT 5+6-(7-3)+4
11
```

Aufgrund der Klammern muß die Maschine zunächst 3 von 7 subtrahieren, ehe sie die übrigen Operationen ausführen kann.

Multiplikation

Die Multiplikation wird durch das Malzeichen (*) ausgedrückt, wie folgendes Beispiel veranschaulicht:

```
PRINT "13 MAL 14 IST " 13*14
13 MAL 14 IST 182
```

Beachten Sie, daß all das, was Sie nach PRINT in Anführungszeichen setzen, genau so ausgegeben wird, wie Sie es schreiben. Dadurch haben Sie die Möglichkeit, die Bedeutung der einzelnen Zahlen anzugeben.
Wenn Sie eine Summe aus mehreren Zahlen mit einer Zahl multiplizieren müssen, beispielsweise in einer Steuerberechnung, dann müssen Sie erneut Klammern verwenden:

```
PRINT (1.26+0.23+4.99)*0.6
.3888
```

Division

Als Divisionszeichen dient der Schrägstrich (/). Wollen Sie eine Summe aus mehreren Zahlen durch eine Zahl dividieren, müssen Sie, wie bei der Multiplikation, darauf

achten, die Summe in Klammern zu setzen. Auf diese Weise wird die Maschine gezwungen, zuerst die Summe auszurechnen. Angenommen, Sie wollen den Durchschnitt aus 5,43 und 67 berechnen, dann geben Sie folgende Zeile ein:

```
PRINT "DURCHSCHNITT IST"; (5+43+67)/3
DURCHSCHNITT IST 38.3333333
```

Bei einem Ausdruck, der nur Multiplikationen und Divisionen enthält, werden die Operationen wieder von links nach rechts ausgeführt:

```
PRINT 1/2*3
1.5
```

Aber:

```
PRINT 1/(2*3)
.166666667
```

Multiplikation und Division befinden sich also auf der gleichen Stufe in der arithmetischen Hierarchie. Die Maschine arbeitet eine Zeichenreihe, in der Multiplikationen und Divisionen gemischt sind, von links nach rechts ab, sofern Klammern nicht eine andere Reihenfolge festlegen; Multiplikation und Division haben gegenüber Addition und Subtraktion Vorrang. Sie werden also zuerst ausgeführt, wie folgendes Beispiel zeigt:

```
PRINT 3+2*6
15
```

Hätte dagegen die Addition Vorrang gegenüber der Multiplikation, so würde die Maschine zuerst addieren und

dann multiplizieren; um dies zu erreichen, müssen Sie Klammern verwenden:

```
PRINT (2+3)*6
30
```

Potenzierung

Eine weitere arithmetische Operation ist die Potenzierung. Rufen Sie sich in Ihr Gedächtnis, daß durch den Ausdruck »a hoch b« a b-mal mit sich selbst multipliziert wird. Auf der Apple-Tastatur bewirkt das Symbol ^ die Potenzierung eines Ausdrucks:

```
PRINT 3^2
9
```

Oder:

```
PRINT 2^3
8
```

Die Potenzierung steht an der Spitze in der Hierarchie der arithmetischen Operationen, sie wird somit vor Multiplikation und Division vorgenommen. Eine Folge von Potenzierungen arbeitet die Maschine wieder von links nach rechts ab.

Übungen

1. Führen Sie folgende Rechnungen unter Verwendung einer PRINT-Anweisung aus:
   a) 457 + 32 + 94
   b) 34 mal 237
   c) 64 geteilt durch 16
   d) 2 hoch 16

2. Bilden Sie den Durchschnitt aus 3.208, −56.007, 3412.76 und −.004.

3. Welche Mehrwertsteuer (bei einem Satz von 14%) müssen Sie bei den folgenden drei Beträgen insgesamt entrichten: DM 23 689, DM 108 544 und DM 14 320?

4. Zeilennummern und Anweisungen

Es gibt zwei verschiedene Möglichkeiten, nach denen Ihr Apple die eingegebenen Daten verarbeiten kann. Wie Sie gesehen haben, führt der Computer jede Anweisung sofort nach dem Betätigen der RETURN-Taste aus, wenn Sie vor die jeweilige Zeile keine Nummer setzen. Stellen Sie dagegen eine Nummer vor das PRINT-Kommando oder vor jedes andere Kommando, so geschieht nichts, wenn Sie die RETURN-Taste drücken; lediglich die Zeile, die Sie gerade eingegeben haben, springt auf Ihrem Monitor um eine Zeile nach oben, um Platz für die nächste einzugebende Zeile zu schaffen. Das System überprüft, ob die Anweisung mit einer Zeilennummer versehen ist und speichert sie dann ab, ohne sie sofort auszuführen; zugleich ordnet es mehrere Anweisungen nach aufsteigender Zeilennummer. Geben Sie lediglich eine Zeilennummer ein und drücken dann die RETURN-Taste, so löscht der Rechner jede bereits vorher eingegebene Anweisung mit dieser Zeilennummer; tippen Sie eine bereits ausgeführte Zeilennummer mit einer neuen Anweisung ein und bedienen dann die RETURN-Taste, so ersetzt das System die alte Anweisung durch die neue mit derselben Zeilennummer.
Wollen Sie wissen, wie Ihr laufendes Programm aussieht, dann tippen Sie das Kommando LIST ein. Die Programmzeilen werden dann in aufsteigender Reihenfolge der Zeilennummern ausgegeben.
Bei Applesoft BASIC können Sie, was in dieser Programmiersprache ziemlich außergewöhnlich ist, so viele BASIC-Anweisungen in eine Zeile schreiben, wie Sie wollen. Anweisungen in derselben Zeile werden durch einen Doppelpunkt (:) voneinander getrennt. Das ist besonders dann nützlich, wenn Sie in einer Zeile sowohl eine Anwei-

sung als auch einen Kommentar dazu (REM) schreiben wollen; der Kommentar dient nur zu Ihrer Information und wird vom Computer ignoriert.
Geben Sie nun folgendes Programm ein und lassen Sie es laufen:

```
10 REM  ERSTES PROGRAMM
20 PRINT "++++++++++++++++++++++++++"
30 PRINT "+ APPLE IST DER GROESSTE +"
40 PRINT "++++++++++++++++++++++++++"
```

Sobald das Applesoft Promptzeichen nach dem Drücken der RETURN-Taste erscheint, tippen Sie das Kommando RUN ein und bedienen erneut die RETURN-Taste. Dann sollte folgendes auf Ihrem Bildschirm erscheinen:

```
RUN
++++++++++++++++++++++++++
+ APPLE IST DER GROESSTE +
++++++++++++++++++++++++++
```

Erhalten Sie diese Ausgabe nicht, dann geben Sie LIST ein und drücken die RETURN-Taste. Ihr vierzeiliges Programm erscheint nun auf dem Bildschirm, und was sich vorher auf dem Bildschirm befand, verschwindet Zeile für Zeile nach oben. Vergleichen Sie die Ausgabe mit der obigen Auflistung des Programms auf Fehler hin.
Während der Rechner Ihr Programm Zeile für Zeile und in der durch die Zeilennummern festgelegten Reihenfolge ablaufen läßt, überprüft er bei der Ausführung jede einzelne Anweisung. Entdeckt er dabei einen Eingabefehler, der die Anweisung für ihn unverständlich werden läßt, so erscheint die Fehlermeldung ?SYNTAX ERROR mit dem Hinweis auf die betreffende Zeilennummer auf dem Bild-

schirm. So erhalten Sie die Information, in welcher Zeile eine Korrektur notwendig ist, um das Programm laufen zu lassen.
Zur Änderung einer Programmzeile geben Sie die entsprechende Zeilennummer sowie die korrekte Fassung der Anweisung ein. Bedienen Sie dann die RETURN-Taste, ersetzt die Maschine die alte durch die neue Zeile.

Übungen

1. Schreiben Sie ein vierzeiliges Programm mit drei Anweisungen pro Zeile. Vergessen Sie dabei nicht, einen Doppelpunkt (:) zwischen den einzelnen Anweisungen zu setzen.
2. Lassen Sie sich Ihr vierzeiliges Programm aus Übung 1 ausgeben. Suchen Sie eine Anweisung, nach deren Löschung Ihr Programm immer noch ablauffähig ist. Löschen Sie diese Anweisung, indem Sie die Zeilennummer, gefolgt von der geänderten Zeile, eintippen. Lassen Sie Ihr Programm ausdrucken, um sicherzustellen, daß alles in Ordnung ist, und lassen Sie es dann laufen.
3. Geben Sie einige Berechnungen mit Hilfe von PRINT mit vorangestellten Zeilennummern ein. Lassen Sie Ihr Programm laufen.
4. Schreiben Sie ein Programm, das nach drei Leerzeilen Ihren Namen und Ihre Adresse einschließlich der Postleitzahl druckt.

5. Variablen und Zuweisungen

Programme gewinnen ihren Wert erst dadurch, indem Sie Variablen anstelle von Zahlen verwenden, weil Sie dann dasselbe Programm mit verschiedenen Zahlen oder Zeichen laufen lassen können. Eine Variable ist nichts anderes als ein Name für einen Speicherplatz, in dem die Daten stehen, mit denen Ihr Programm arbeiten soll. Ihr Applesoft BASIC kennt drei Arten von Variablen, die drei verschiedenen Datentypen entsprechen.

Abgesehen von speziellen Zwecken verwendet man zum Rechnen mit Zahlen Gleitpunktzahlen mit neun Stellen hinter dem Komma. Den Gleitpunktzahlen entsprechen Variablennamen, die nur aus Buchstaben und Ziffern bestehen, und deren erstes Zeichen ein Buchstabe sein muß.

Versuchen Sie sich an diesem Programm:

```
10 REM  ERSTES PROGRAMM MIT VARIABLEN
20 V=3 : X=5 : W=-9
30 PRINT V+X-W, "HURRA"
```

Sie sollten folgende Ausgabe erhalten, nachdem Sie das Kommando RUN eingegeben haben:

```
17              "HURRA"
```

Ist dies nicht der Fall, dann lassen Sie Ihr Programm ausgeben und suchen die Fehler. In Zeile 20 stehen Doppelpunkte (:), weil Sie mehr als eine Anweisung in eine Zeile geschrieben haben.

Sie können Variablen auch benutzen, um Zeichenreihen zu speichern wie in folgendem Beispiel:

```
10 REM   STRINGVARIABLEN
20 A$="HURRA"
30 PRINT "***";A$;"***";A$;"***"
```

In Zeile 20 sind einige Punkte bemerkenswert. Zum einen endet der Name der Variablen mit einem Dollarzeichen; damit wird angezeigt, daß sie für eine Zeichenreihe steht. Man spricht in diesem Fall von einer »Stringvariablen«. Beachten Sie zum anderen, daß die Zeichenreihe selbst, HURRA, von Anführungszeichen eingeschlossen ist. Immer wenn Sie einer Stringvariablen eine Zeichenreihe zuweisen, müssen Sie die Zeichenreihe in Anführungszeichen setzen.
Der Rechner benutzt nur die ersten zwei Zeichen eines Variablennamens, um die Variable eindeutig zu identifizieren. Sie können jedoch längere Variablennamen benutzen, um sich leichter daran zu erinnern, wofür die Variablen stehen. Ein Variablenname darf außerdem keines der von Applesoft BASIC verwendeten Schlüsselworte, wie RUN oder LIST, enthalten. Wo möglich, sollten Sie daher einen Namen verwenden, der aus einem oder zwei Zeichen besteht.
Betrachten Sie dieses Programm:

```
10 REM   ARITHMETISCHE ZUWEISUNGEN
20 X=(5+6)/3 : Y=2/3
30 A=X+Y
40 A$="DIE ANTWORT LAUTET "
50 PRINT A$;A
```

Immer wenn eine Anweisung mit einem Variablennamen beginnt, dem ein Gleichheitszeichen folgt, handelt es sich um eine Zuweisung (»Assignement«). Ihre Funktion besteht darin, den Ausdruck rechts vom Gleichheitszeichen zu nehmen und ihn in den Speicherplatz zu bringen, der durch den Ausdruck links davon markiert wird. Bei

Applesoft BASIC können Sie der linksstehenden Variablen das Kommando LET voranstellen. Aus diesem Grund werden Zuweisungen auch LET-Anweisungen genannt.

Übungen

1. Weisen Sie den Variablen P, C und E die Werte 3.14159, 181156 und .007 zu und lassen Sie dann das Produkt dieser drei Zahlen ermitteln. Schreiben Sie dazu je eine Anweisung pro Zeile, also insgesamt vier Zeilen. Lassen Sie dann Ihr Programm ausdrucken und anschließend laufen.
2. Weisen Sie der Zeichenreihenvariable Z$ den Wert ZORRO zu. Drucken Sie das Wort dann fünfmal in eine Zeile, wobei jeweils drei Leerzeichen dazwischen stehen sollen.
3. Bilden Sie ein großes Z auf dem Bildschirm unter Benutzung des Wortes ZORRO aus Übung 2. (Nur für Fortgeschrittene.)
   Hinweis: Sie benötigen dazu 24 Druckanweisungen.

6. Die Eingabe von Daten über die Tastatur

Bei den Programmen, die Sie bis jetzt ausprobiert haben, war es nicht erforderlich, noch irgendetwas zu tun, sobald das Programm erst einmal gestartet wurde. Nur wenn das Programm irgendwelche Daten enthielt, mußten Sie diese in das Programm schreiben.
Sehr oft weiß ein Programmierer nicht, welche Zahlen benötigt werden, wenn das Programm läuft. Deshalb muß ein Hilfsmittel verwendet werden, um die tatsächlichen Daten während der Ausführung des Programms eingeben zu können. BASIC kennt zu diesem Zweck die INPUT-Anweisung. Damit Sie sich daran gewöhnen, geben Sie das folgende Programm ein und lassen es laufen.

```
10 REM   ERSTES PROGRAMM MIT EINGABE
20 INPUT "GEBEN SIE EINE ZAHL AN ";N
30 PRINT "IHRE ZAHL IST ";N
```

Wenn Sie das Programm laufen lassen, erscheint die Zeile GEBEN SIE EINE ZAHL AN auf dem Bildschirm, und Sie sehen, daß der Kursor blinkt und auf die Eingabe Ihrer Zahl N wartet. Wenn Sie etwas anderes als eine Zahl eintippen, dann erhalten Sie eine ?REENTER-oder auch eine ?EXTRA IGNORED-Meldung. Experimentieren Sie mit diesem Programm herum, und machen Sie dabei verschiedene Eingabefehler. Beobachten Sie, was geschieht. Sie können auch Strings eingeben:

```
10 REM   EINGABE VON STRINGS
20 INPUT "WIE LAUTET IHR VORNAME ";F$
30 INPUT "WIE LAUTET IHR NACHNAME ";L$
40 PRINT "IHR GANZER NAME LAUTET ";F$;
   " ";L$
```

Es ist in BASIC auch erlaubt, mehrere Daten in einer Zeile aufzuführen:

```
10 REM   EINGABE MEHRERER DATEN
20 INPUT A, A$, B
```

Wenn Sie dieses Programm laufen lassen, erscheint ganz links ein Fragezeichen, hinter dem der Kursor auf die Eingabe einer Zahl für die Variable A wartet. Tippen Sie etwas anderes als eine Zahl ein (versuchen Sie das), erhalten Sie die Meldung ?REENTER; geben Sie schließlich eine Zahl ein, wird diese in dem durch A markierten Speicherplatz abgelegt.
Der Rechner wird Sie dann mit einem doppelten Fragezeichen (??) zu einer weiteren Eingabe veranlassen, wenn Sie noch nicht die Werte für alle in der INPUT-Anweisung stehenden Variablen eingetippt haben.
Um eine INPUT-Anweisung mit einem einzigen Drücken der RETURN-Taste zu erledigen, müssen Sie Ihre Eingabedaten durch Kommas voneinander trennen und alle Strings in Anführungszeichen setzen. Geben Sie

```
?5, "HURRA",6
```

ein, dann ist die obige Zahl 20 nach dem einmaligen Drücken der RETURN-Taste erledigt.
Es ist in der Regel am besten, jeweils nur eine Variable einzugeben und bei jeder Eingabe eine Prompt-Meldung zu verwenden.

Übungen

1. Schreiben Sie ein Programm, das die Multiplikationskenntnisse einer Person testet. Bitten Sie sie, die folgenden Multiplikationen auszuführen:
   2 * 5, 5 * 9, 43 * 17 und 856 * 222. Fragen Sie die Person jeweils nach einem Ergebnis und vergleichen Sie

die Auskunft mit dem korrekten Wert. Wenn beide übereinstimmen, dann geben Sie SEHR GUT aus und gehen zur nächsten Frage über. Ist die Antwort falsch, drucken Sie NEIN, DIE KORREKTE ANTWORT IST, gefolgt von der richtigen Antwort. Dann gehen Sie zur nächsten Frage über. Wenn alle Fragen gestellt und beantwortet sind, informieren Sie die Person darüber, wie viele Antworten richtig waren.
Hinweise: Fangen Sie mit einem einfachen Programm an, das lediglich die erste Frage stellt. (Wenn das läuft, können Sie weitermachen.) Wiederholen Sie die Struktur, die Sie bei der ersten Frage gewählt haben, noch dreimal. Vielleicht möchten Sie Unterprogramme für die Meldungen an den Benutzer verwenden, weil diese für jede Frage gleich sind. Angenommen, Sie verwenden in Ihrem Programm die folgenden Zeilen:

```
1000 PRINT "SEHR GUT"
1010 RETURN
```

Dann können Sie immer, wenn Sie SEHR GUT drucken wollen, folgendes Kommando schreiben:

```
130 GOSUB 1000
140 REM   PROGRAMM WIRD HIER FORTGESETZT
```

Das Programm wird zur Zeile 1000 springen und damit beginnen, alle Anweisungen, die von dort an stehen, auszuführen. Wenn es auf eine RETURN-Anweisung stößt, springt es sofort auf die dem Kommando GOSUB folgende Anweisung zurück.

2. Fragen Sie jemanden, ob er schon einmal von schwarzen Löchern gehört hat. Fassen Sie seine Antwort unter die Variable A$ und testen Sie sie. Wenn sie J oder JA ist, drucken Sie OH, SIE WISSEN EINE MENGE und

halten an. Wenn A$ den Wert N oder NEIN besitzt, dann drucken Sie SIE SIND FASZINIEREND und halten an. Bei jeder anderen Antwort geben Sie BITTE WIEDERHOLEN SIE aus und stellen die Frage erneut.

7. Datenspeicherung im Programm

Es gibt noch eine weitere Möglichkeit, um Ihr Programm während der Ausführung mit Daten zu versorgen. Bei der in diesem Kapitel beschriebenen Methode geben Sie die Daten in das Programm selbst ein, wobei Sie speziell dafür vorgesehene DATA-Anweisungen verwenden. Angenommen, Sie listen die Namen von vier Personen auf, dazu deren Alter und ihre Körpergröße in Zentimetern. Geben Sie also zunächst folgende Daten ein:

```
10 REM   PROGRAMM MIT DATA-ANWEISUNGEN
20 DATA "GEORG SCHMIDT", 23,172
30 DATA "MONIR AFGAJAN", 8,138
40 DATA "QUAN TRAN", 67, 168
50 DATA "MARCY WU", 34,194
```

Zunächst lassen Sie diese DATA-Anweisungen lesen und die Daten dann drucken. Damit Sie dazu nicht vier verschiedene Druckanweisungen schreiben müssen, soll das Programm immer wieder zu ein und derselben READ-Anweisung zurückspringen und, nachdem alle Daten gelesen sind, anhalten. Versuchen Sie es:

```
60 READ N$,A,H
70 PRINT N$,A,H
80 GOTO 60
```

Sie erhalten die Meldung ?OUT OF DATA ERROR IN 60, und der Kursor wartet auf Ihre nächste Anweisung. Beachten Sie, daß die einander entsprechenden Datenelemente auf dem Bildschirm untereinander angeordnet

erscheinen, da Sie Kommas zwischen die zu druckenden Elemente in Zeile 70 gesetzt haben.
Man hätte auch sämtliche Daten in einer einzigen langen DATA-Anweisung angeben oder auch eine eigene DATA-Anweisung für jedes Datenelement verwenden können.

Die Anweisungen READ und DATA arbeiten eng miteinander zusammen und unterliegen dabei den folgenden einfachen Regeln:

1. Sämtliche Datenelemente, die in den DATA-Anweisungen Ihres Programms vorkommen, werden zu einer einzigen Liste von Datenelementen zusammengefaßt. Ihre Reihenfolge entspricht dabei der aufsteigenden Reihenfolge der Zeilennummern der DATA-Anweisungen.

2. Jedesmal, wenn eine irgendwo im Programm stehende READ-Anweisung ein Datenelement lesen muß, dann erfaßt sie das erste verfügbare Datenelement, das vom laufenden Programm noch nicht gelesen wurde. Beim ersten Datenelement, das eingelesen wird, handelt es sich um das erste Datenelement derjenigen DATA-Anweisung, die die niedrigste Zeilennummer trägt.

Sie lernen weiter unten noch eine weitere Möglichkeit kennen, wie man alle vier DATA-Anweisungen einlesen kann, ohne vier INPUT-Anweisungen schreiben zu müssen und auch ohne aus einer Schleife nur mittels einer Fehlermeldung herauszukommen; das geschieht mit Hilfe einer FOR... NEXT-Schleife, die in Kapitel 12 besprochen wird. Für den Augenblick reicht es aus, dem Programm einfach die beiden folgenden Anweisungen hinzuzufügen.

```
55  FOR I=1 TO 4
80  NEXT I
```

Mit Ausnahme von Zeile 80 bleiben die bereits existierenden Zeilen erhalten; die alte Zeile 80 wird durch die neue

Zeile 80 ersetzt. Zeile 55 kommt neu hinzu. Lassen Sie sich nun das ganze Programm mit Hilfe von LIST ausgeben, und starten Sie es dann mit Hilfe des Kommandos RUN. Sie sollten folgende Ausgabe erhalten:

```
]LIST

10  REM   PROGRAMM MIT DATA-ANWEISUNGEN
20  DATA  "GEORG SCHMIDT",23,172
30  DATA  "MONIR AFGAJAN",8,138
40  DATA  "QUAN TRAN",67,168
50  DATA  "MARCY WU",34,194
55  FOR I = 1 TO 4
60  READ N$,A,H
70  PRINT N$,A,H
80  NEXT I

]RUN
GEORG SCHMIDT   23       172
MONIR AFGAJAN   8        138
QUAN TRAN       67       168
MARCY WU        34       194
```

8. Ein Einkaufslisten-programm

Vielleicht haben Sie in einem Supermarkt schon einmal jemanden gesehen, der einen Taschenrechner in der Hand hielt, die Preise eingab und auf diese Weise die Preissumme der Waren berechnete. Ein Einkaufslisten-programm kann sehr hilfreich sein, wenn es die Bezeichnung des gekauften Artikels direkt neben dem Preis ausgibt. Bei dem Programm, das wir im folgenden entwickeln werden, müssen Sie den Namen der zu kaufenden Ware eingeben, die gekaufte Stückzahl, den Preis pro Stück und einen Code, der besagt, ob dazu noch Steuern zu entrichten sind oder nicht (0 = keine Steuer, 1 = Steuer). Verwenden Sie eine DATA-Anweisung pro Ware, beispielsweise in folgender Form:

```
100 DATA "MILCH",3,1.11,0
```

Diese Anweisung sagt dem Programm, daß wir drei Tüten Milch kaufen, und zwar zum Preis von 1.11 DM pro Tüte, und daß auf Milch keine Steuer erhoben wird. Sie sollten bei Ihren DATA-Anweisungen darauf achten, daß die Zeilennummern erst bei 100 anfangen, damit Sie noch Platz für Einfügungen haben. Die letzte DATA-Anweisung muß eine »leere« Warenbezeichnung und jeweils Null für die übrigen Angaben enthalten. Wenn das Programm auf eine solche Anweisung stößt, wird es die Summen ausgeben und anhalten.

Was leistet das Programm nun eigentlich? Zum einen gibt es jede Ware, die eingelesen wird, aus und multipliziert die Stückzahl mit dem Preis pro Stück, um den Gesamtpreis für die jeweilige Ware zu berechnen. Werden auf die Ware noch Steuern aufgeschlagen, dann können Sie zum andern sehen, welcher Steuerbetrag für die

jeweilige Ware zu zahlen ist; das angeführte Beispiel geht von einem Steuersatz von 6% aus. Sind alle Waren gelesen und wieder ausgedruckt, erhalten Sie außerdem den Gesamtpreis aller Waren ohne Steuern, ferner den Gesamtbetrag der zu entrichtenden Steuern und natürlich den insgesamt zu zahlenden Betrag.

Sofern Sie sich bereits zutrauen, ein solches Programm selbständig zu schreiben, dann versuchen Sie es. Andernfalls studieren Sie das folgende Programm, es stellt eine mögliche Lösung dar:

```
]LIST

 10  REM ***EINKAUFSLISTE***
 20  READ N$,Q,P,T
 30  IF N$ = "" THEN  GOTO 110
 40  C = Q * P
 50  TC = TC + C
 60  IF T = 1 THEN TX = C * .06
 70  IF T = 0 THEN TX = 0
 80  TT = TT + TX
 90  PRINT  Q,N$,C,TX
100  GOTO 20
110  PRINT "GESAMTPREIS= ";TC
112  PRINT "GESAMTE STEUER= ";TT
114  PRINT "ZU ZAHLENDER BETRAG ";TC
      + TT
120  DATA "BROT",2,1.56,0
130  DATA "BIER",2,2.54,1
140  DATA "SHAMPOO",1,3.66,1
150  DATA "KATZENFUTTER",14,.33,0
200  DATA "MILCH",3,1.11,0
210  DATA "EIER",1,.75,0
300  DATA "",0,0,0
```

```
]RUN

2     BROT           3.12
0
2     BIER           5.08
.3048
1     SHAMPOO        3.66
.2196
14    KATZENFUTTER
4.62  0
3     MILCH          3.33
0
1     EIER           .75
0
GESAMTPREIS= 20.56
GESAMTE STEUER= .5244
ZU ZAHLENDER BETRAG: 21.0844
```

9. Arithmetische Ausdrücke

Sie erinnern sich sicherlich an den in Kapitel 3 gegebenen Hinweis, daß zwischen den verschiedenen arithmetischen Operationen eine Hierarchie besteht, wobei Operationen auf gleicher Stufe von links nach rechts abgearbeitet werden:

1. Potenzierung
2. Multiplikation und Division
3. Addition und Subtraktion

Beschäftigen Sie sich näher mit der Programmierung mathematischer Ausdrücke, müssen Sie meist ziemlich komplizierte Gleichungen in Applesoft BASIC übersetzen. Hier ein Beispiel für die Art von Formeln, mit denen Sie es dabei vermutlich zu tun bekommen:

$$P1 = \frac{\frac{R}{1200} \cdot \left(1 + \frac{R}{1200}\right)^{M} \cdot L}{\left(1 + \frac{R}{1200}\right)^{M} - 1}$$

In diesem Fall ist es vorteilhaft, den Ausdruck in seinen Zähler Z1 und in seinen Nenner N1 zu zerlegen:

$$Z1 = \frac{R}{1200} \cdot \left(1 + \frac{R}{1200}\right)^{M} \cdot L$$

$$N1 = \left(1 + \frac{R}{1200}\right)^{M} - 1$$

Auf diese Weise können Sie jeden der beiden Ausdrücke gesondert auflösen. Der Nenner sieht leichter aus, fangen Sie also damit an. Der Ausdruck läßt sich in folgender Form darstellen:

```
D1=(1+R/1200) M-1
```

Um diese Umformungen zu erhalten, müssen Sie stets die Hierarchie berücksichtigen, nach der die Maschine vorgeht. Weil die Potenzierung auf der obersten Stufe der Hierarchie steht, erhebt sie alles, was innerhalb der Klammern steht, in die M-te Potenz und zieht dann 1 ab. Der Ausdruck innerhalb der Klammern macht es erforderlich, daß man zunächst R durch 1200 teilt und dann 1 zum Ergebnis hinzuaddiert. Dabei wird die Division vor der Addition ausgeführt.
Der Zähler erfordert eine geringfügig andere Behandlung. Da der Teilausdruck »R dividiert durch 1200« zweimal im Zähler auftaucht, ist es naheliegend, für ihn einen eigenen Ausdruck einzuführen und sich somit eine doppelte Eingabe zu ersparen:

```
F=R/1200
```

Durch eine Zuweisung dieser Art gewinne Sie Zeit beim Programmieren. Der Zähler erhält somit die folgende Darstellung:

```
Z1=F*(1+F)^M*L
```

Damit sieht das vollständige Programmstück zur Berechnung von P1 wie folgt aus:

```
100 F=R/1200
110 N1=(1+F)^M-1
120 Z1=F*(1+F)^M*L
130 P1=Z1/N1
```

Übungen

1. Angenommen, die Variablen A, B und C haben die folgenden Werte: A = 3.28, B = –6.43, C = 1.087. Schreiben Sie arithmetische Ausdrücke, die den folgenden Formeln entsprechen; testen Sie Ihre Ausdrücke dadurch, daß Sie die entsprechenden Programme laufen lassen.

   (a) $\frac{A + B}{C}$ (b) $\frac{A = B^2}{A - B}$

   (c) $\frac{A^{(B^2-4C)} + 2AB + B^5}{C^2 - 9A + 1}$

2. Ändern Sie die Zeilen 100 bis 130 in zeitsparender Weise ab. Beachten Sie, daß bei der bisherigen Lösung ein und dieselbe Berechnung zweimal ausgeführt wird.
3. Der Abweichungsfaktor V des Raumschiffs Astrologos ist das Produkt aus der Abschirmenergie E, der Geschwindigkeit des Raumschiffs G und dem Gravitätskoeffizienten K. Der Gravitätskoeffizient ist die dritte Potenz von PI (PI = 3.14159) plus die Hamburgkonstante .06351. Schreiben Sie einen einzelnen arithmetischen Ausdruck, durch den der Abweichungsfaktor berechnet wird. Testen Sie Ihren Ausdruck mit diesen Werten: E = 4.238, V = 115.233 (Kilometer pro Sekunde). Wie groß ist der Abweichungsfaktor?

10. Logische Ausdrücke

Im letzten Abschnitt haben Sie gesehen, daß der Wert eines arithmetischen Ausdrucks eine Zahl ist, und daß die Maschine diese Zahl dadurch erhält, daß sie die aktuellen Werte der Variablen in den Ausdruck einsetzt.

Ebenso nützlich und ähnlich aufgebaut wie arithmetische Ausdrücke sind logische Ausdrücke; sie setzen sich aus logischen Variablen (oder Konstanten) sowie logischen Operationen aus diesen Variablen und Konstanten zusammen. Jeder logische Ausdruck, und das gilt auch für Konstanten und Variablen, kann nur die Werte 0 (falsch) und 1 (wahr) annehmen. Zu jedem Zeitpunkt ist ein gegebener logischer Ausdruck also entweder wahr (Wert = 1) oder falsch (Wert = 0).

In der Regel dienen logische Ausdrücke zum Vergleich zweier Zahlen. Es gibt insgesamt sechs Möglichkeiten solcher Vergleiche:

BASIC-Ausdruck	Mathematische Bedeutung
A=B	A ist gleich B
A<>B	A ist ungleich B
A>B	A ist größer als B
A>=B	A ist größer oder gleich B
A<B	A ist kleiner als B
A<=B	A ist kleiner oder gleich B

Mit diesem Wissen ausgerüstet, können Sie einiges auf Ihrem Rechner ausprobieren. Sie wissen beispielsweise, daß 3 kleiner als 5 ist. Testen Sie den Rechner, ob er es auch weiß, indem Sie eingeben:

```
PRINT 3<5
```

Üben Sie das, indem Sie verschiedene Ausdrücke dieser Art mittels der Maschine überprüfen.
Logische Variablen können Sie mit Hilfe logischer Operationen, wie AND, OR oder NOT, miteinander verknüpfen.
Die Anweisung

```
10 IF 3<2 OR 4>3 THEN PRINT "HALLO"
```

gibt HALLO aus, weil der zweite Teil des Ausdrucks, also 4 größer 3, wahr ist, obgleich der erste Teil falsch ist. Die Verbindung zweier Ausdrücke durch OR verlangt, daß mindestens einer der beiden Ausdrücke den Wert »wahr« besitzt, damit der gesamte Ausdruck als wahr gilt. Daher erhalten Sie für den Ausdruck

```
20 IF 3<2 OR 4>5000 PRINT "HALLO"
```

keinerlei Angabe.

Eine weitere logische Operation, AND, verlangt, daß beide Teilausdrücke wahr sind, damit der Gesamtausdruck den Wert »wahr« annimmt.
Welcher Wahrheitswert wird sich für den folgenden Ausdruck ergeben?

```
PRINT ( 3<4 AND 4<=3 ) OR 3^2<>8
```

Geben Sie den Ausdruck ein, drücken Sie die RETURN-Taste und stellen Sie fest, ob Sie recht haben.

Übungen

1. Schreiben Sie ein Programm, das den Benutzer nach Werten für die Variablen x, y und z fragt und dann damit den folgenden logischen Ausdruck auswertet:

```
x^2<=y+1 or x<-47 or x^2+y^2<=z^2
```

   Das Programm soll WAHR drucken, wenn der logische Ausdruck den Wert »wahr«, und FALSCH, wenn der logische Ausdruck den Wert »falsch« besitzt.

2. Schreiben Sie einen logischen Ausdruck, der wahr ist, wenn 3 mal n zum Quadrat größer oder gleich 57 minus x ist.

3. Nehmen Sie an, der Benutzer hätte gerade die Antwort auf eine Ja/Nein-Frage eingegeben. Schreiben Sie eine IF...THEN-Anweisung, die zur Anweisung 340 verzweigt, wenn die eingegebene Zeichenreihe A$ den Wert J oder JA hat.

4. Schreiben Sie ein Programm, bei dem ein Zinssatz in Prozent und ein Kapital in DM eingegeben werden müssen. Wenn der Zinssatz über 15 oder unter 10% liegt oder wenn das Kapital 25000 DM übersteigt, dann soll das Programm die Meldung BITTE UEBERPRUEFEN SIE DIE ANGABEN ZUR ANLEIHE ausgeben und anhalten. Andernfalls soll die Ausgabe ANLEIHE SCHEINT IN ORDNUNG ZU SEIN erscheinen und das Programm ebenfalls stoppen.

11. IF...THEN-Anweisungen

Jedes Computerprogramm muß Entscheidungen treffen: Wenn das und das wahr ist, dann tue dieses, andernfalls tue jenes. Ohne diese Fähigkeit verarbeitet ein Programm entweder nichts oder es bleibt in einer endlosen Schleife stecken.
Die IF...THEN-Anweisung stellt in BASIC die gebräuchlichste Form einer Anweisung dar, die ein Programm in die Lage versetzt, Entscheidungen zu treffen. Eine IF...THEN-Anweisung besteht aus dem Wort IF, dem ein logischer Ausdruck, das Wort THEN sowie eine oder mehrere weitere Anweisungen in derselben Zeile folgen. Hier ein Beispiel:

```
IF 3<5 THEN PRINT "RICHTIG"
```

Weist der logische Ausdruck zum Zeitpunkt der Ausführung der IF-Anweisung den Wert »wahr« auf, so führt die Maschine die Anweisung, die unmittelbar hinter dem Wort THEN steht, aus. Folgen mehrere Anweisungen in der Zeile der IF-Anweisung, dann werden diese nacheinander ausgeführt. Besitzt der logische Ausdruck dagegen den Wert »falsch«, so fährt das Programm mit der nächsten Anweisung fort, die mit einer Zeilennummer versehen ist. Als Ausgabe des Programms

```
10 IF 3<>3 THEN PRINT "HIP":PRINT "HIP"
20 PRINT "HURRA"
```

erhalten Sie einfach den Ausdruck »HURRA«.

IF...THEN-Anweisungen werden sehr häufig verwendet, um festzustellen, ob die Abarbeitung einer Schleife beendet werden kann oder nicht. Im folgenden Programm

stoppen Sie durch eine IF...THEN-Anweisung das Programm, nachdem 10 ausgegeben worden ist.

```
10 N=N+1
20 PRINT N
30 IF N<10 THEN GOTO 10
40 STOP
```

Dieses Programm nutzt den Umstand, daß sämtliche Variablen zu Beginn der Ausführung eines Programms mit dem Wert Null besetzt werden. Daher ist der erste Wert von N, der ausgegeben wird, N = 1.

Der logische Ausdruck, der auf das Wort IF folgt, kann so kompliziert sein, wie er will, solange Sie ihn selbst noch verstehen; im allgemeinen kommt man aber mit sehr einfachen Ausdrücken gut aus.

Übungen

1. Schreiben Sie ein kurzes Programm, das den Benutzer nach seiner Lieblingszahl fragt. Ist diese kleiner als 47, dann soll das Programm TUT MIR LEID, DIE IST ZU KLEIN drucken und anhalten; ist die eingegebene Zahl aber 47 oder größer, dann soll TUT MIR LEID, DIE IST ZU GROSS ausgegeben werden. Achten Sie darauf, daß Ihr Programm nur eine der beiden Meldungen druckt.

Lösung:

```
10 INPUT "WAS IST IHRE LIEBLINGSZAHL?";f
20 IF f<47 PRINT "TUT MIR LEID, DIE IST
   ZU KLEIN":
STOP
30 PRINT "TUT MIR LEID, DIE IST ZU GROSS"
```

2. Fragen Sie den Benutzer nach seiner Hausnummer. Drucken Sie GERADE, wenn es sich um eine gerade Zahl, und UNGERADE, wenn es sich um eine ungerade Zahl handelt.

Hinweis: Um herauszufinden, ob eine bestimmte Zahl N gerade ist, teilt man sie durch 2 und überprüft, ob es sich dabei um eine ganze Zahl handelt. Machen Sie das mit Hilfe des folgenden logischen Ausdrucks:

```
N/2=INT(N/2)
```

3. Lassen Sie die Maschine in Zehnerschritten von 10 bis 300 zählen; sie soll also 10, 20, 30, 40 usw. ausgeben. Lösung:

```
10 N=N+10
20 PRINT N
30 IF N<300 THEN GOTO 10
40 STOP
```

4. Fragen Sie den Benutzer nach zwei Zahlen. Wenn das Quadrat der ersten Zahl größer ist als das Doppelte der zweiten Zahl, dann soll JA, andernfalls NEIN ausgegeben und angehalten werden.

12. FOR...NEXT-Schleifen

Die Notwendigkeit, bestimmte Schritte ständig zu wiederholen, stellt die Grundlage beinahe jeder Arbeit dar, ganz gleich, ob es sich dabei um das Bilden einer Summe, um den Aufbau einer Eiweißverbindung oder um das Mähen des Rasens handelt. Tatsächlich ist der Begriff der Maschine ja sehr eng verbunden mit der unablässigen Wiederholung eines Vorgangs.
Programmierer sprechen, wenn eine Sache immer wieder getan werden soll, von einer Schleife. Beim Bilden einer Summe aus vielen Zahlen werden in einer Schleife ständig einzelne Zahlen zur laufenden Zwischensumme dazugezählt. Beim Aufbau einer Eiweißverbindung gliedert sich jeweils eine weitere Aminosäure dem wachsenden Molekül an. Beim Mähen des Rasens besteht der einzelne Arbeitsgang aus dem Mähen einer Bahn und einer Drehung.
In unserem Zusammenhang ist eine Schleife eine Folge von BASIC-Anweisungen, die von einer FOR-Anweisung am Anfang und einer NEXT-Anweisung am Ende eingeschlossen werden. Die durch die FOR...NEXT-Anweisung eingegrenzten Programmzeilen bezeichnet man auch als Schleifenrumpf oder Schleifenkörper. Versuchen Sie einmal folgendes Schleifenprogramm:

```
10 REM    ERSTE FOR...NEXT-SCHLEIFE
20 FOR N=1 TO 24
30 PRINT "ALLES BEREIT"
40 NEXT N
RUN
```

Wenn Sie Zeile 40 betrachten, dann sehen Sie hinter dem Wort NEXT eine Variable, in diesem Fall N. Es handelt sich

um dieselbe Variable, die in Zeile 20 hinter dem Wort FOR steht. Sie wird als Schleifenindex bezeichnet. Sein Wert wird während der Ausführung der Schleife wiederholt geändert und geprüft.
Im folgenden ist dieses Programm noch einmal mit einigen zusätzlichen Anmerkungen abgedruckt, um die im Zusammenhang mit FOR...NEXT-Schleifen auftretenden Begriffe zu verdeutlichen.

```
10 REM      ERSTE FOR...NEXT-SCHLEIFE
```

Schleifenindex (→ I)
Startwert (→ 1)
Endwert (→ 24)

```
20 FOR I = 1 TO 24
```

Alle Anweisungen, die zwischen FOR und NEXT stehen, bilden den Schleifenrumpf.

```
30 PRINT "ALLES BEREIT"
```

Bei Applesoft BASIC muß der Schleifenindex an dieser Stelle nicht unbedingt geschrieben werden.

```
40 NEXT I
```

Die Anweisung markiert das Ende der Schleife.

Applesoft BASIC besetzt den Schleifenindex zuerst mit dem Startwert und führt dann die im Schleifenrumpf stehenden Anweisungen sofort aus, wobei der Wert des Schleifenindex nicht verändert wird. Bei der Ausführung der NEXT-Anweisung vergleicht die Maschine den Wert des Schleifenindex mit dem Endwert, der unmittelbar hin-

ter dem Wort TO der FOR-Anweisung steht. Erreicht oder überschreitet der Schleifenindex den Endwert, beendet der Computer die Ausführung der Schleife und setzt die weitere Abarbeitung des Programms mit der Anweisung fort, die auf die NEXT-Anweisung folgt. Hat der Wert des Schleifenindex jedoch den Endwert noch nicht erreicht, wird eine Schrittweite zum Schleifenindex hinzugezählt; die Maschine setzt die Ausführung des Programms mit der Anweisung fort, die der FOR-Anweisung unmittelbar folgt, also mit der ersten Anweisung des Schleifenrumpfs. Wenn, wie in Zeile 20, keine Schrittweite angegeben ist, dann wird 1 als Schrittweite angenommen. Andernfalls sind hinter dem Endwert das Wort STEP und ein Wert für die Schrittweite notwendig. Die Schrittweite kann sowohl positiv als auch negativ sein.
Die allgemeine Form einer FOR...NEXT-Schleife sieht daher folgendermaßen aus:

```
FOR <SCHLEIFENINDEX>=<STARTWERT>TO
<ENDWERT> STEP
<SCHRITTWEITE>
<SCHLEIFENRUMPF>
NEXT <SCHLEIFENINDEX>
```

Die Angabe des Schleifenindex hinter dem Wort NEXT kann bei Applesoft BASIC unterbleiben. In diesem Fall nimmt die Maschine an, daß die NEXT-Anweisung zur vorhergehenden FOR-Anweisung gehört.

Übungen

1. Verwenden Sie eine FOR...NEXT-Schleife, um tausendmal DIE JEDIRITTER KOMMEN zu drucken. Drükken Sie gleichzeitig S und die CTRL-Taste, um die Ausgabe abzubrechen oder um sie weiterlaufen zu lassen. Drücken Sie RESET, um das Programm anzuhalten, wenn Sie etwas anderes machen wollen.

2. Betrachten Sie folgenden Shanty:

```
OH, WHISKY BRAUCHT EIN GUTER MANN
OH, WHISKY, JOHNNY
```

Lassen Sie den Shanty fünfmal drucken. Verwenden Sie zwei FOR...NEXT-Schleifen, wobei die äußere von 1 bis 5 läuft, die innere von 1 bis 7. Jede Refrainzeile OH, WHISKY, JOHNNY soll in einer eigenen Zeile stehen.

3. Wie viele der ganzen Zahlen von 1 bis 1000 (also 1, 2, 3, 4...) sind durch 2, 5 oder 17 teilbar?
   Hinweis: Verwenden Sie innerhalb einer FOR ... NEXT-Schleife, die von 1 bis 1000 läuft, eine große IF-Anweisung, die, wenn der logische Ausdruck den Wert »wahr« besitzt, einen Zähler um Eins erhöht. Der logische Ausdruck hinter dem Wort IF muß aus drei Ausdrücken bestehen, die durch zwei OR-Anweisungen voneinander getrennt sind. Der erste dieser Ausdrücke liefert den Wert »wahr«, wenn die Zahl durch 2 teilbar ist. Ein Ausdruck, mit dem man das feststellen kann, ist N/2 = INT (N/2).

4. Wie viele der ganzen Zahlen von 1 bis 1000 sind Quadratzahlen?
   Hinweis: Verwenden Sie die Funktion SQR(N), um die Quadratwurzel aus N zu ziehen, und die Funktion INT, um festzustellen, ob es sich bei dem Ergebnis um eine ganze Zahl handelt.

13. Darlehensrückzahlung

Fast jeder Mensch leiht sich heutzutage von Zeit zu Zeit Geld, wofür es die verschiedensten Darlehensformen gibt. Im Mittelpunkt eines jeden Darlehens steht natürlich der Betrag, der ausgeliehen wird und den man als Kapital bezeichnet; das untenstehende Beispiel verwendet den Buchstaben B zur Darstellung dieser Größe:

```
20 INPUT "WIEVIEL WOLLEN SIE LEIHEN?";B
```

Die nächste Frage gilt der Höhe des Zinssatzes:

```
30 INPUT "DER JAEHRLICHE ZINSSATZ?";I
```

Angenommen, Sie kennen die Höhe des jährlichen Zinssatzes I in Prozent, der beispielsweise bei 12,6% liegt; wie erhalten Sie dann jene Zahl, mit der Sie den Kapitalbetrag multiplizieren müssen, um die Höhe des jährlichen Zinssatzes zu erhalten? Sie dividieren dazu den Betrag des jährlichen Zinssatzes einfach durch 100. So bedeutet 57% von einem Betrag, daß wir den Betrag mit 0.57 multiplizieren müssen. Berechnen Sie die Zinsen täglich, denn die meisten Banken machen es ebenso. Sobald Sie den jährlichen Zinssatz aus Zeile 30 wissen, können Sie ihn ebensogut in den täglichen Zinsfaktor umwandeln:

```
40 D = (I/365)*.01
```

Dieser Wert von D, multipliziert mit dem Kontostand am Morgen des Tages, liefert den Zinsbetrag, der gezahlt werden muß, um den Betrag für einen Tag zu leihen. Erfolgt die Bezahlung der Zinsen für einen Tag nicht, so werden sie zum geschuldeten Betrag hinzugezählt, so

daß sich der am folgenden Tag zu entrichtende Betrag entsprechend erhöht. Weiterhin benötigt das Programm Angaben über den Rückzahlungsplan; sie können in folgender Weise bereitgestellt werden:

```
50 INPUT "WIEVIELE ZAHLUNGEN PRO JAHR?";N
60 INPUT "HOEHE DER ZAHLUNGEN?";P
```

Jetzt verfügt das Programm über sämtliche Informationen, um eine Rückzahlungstabelle auszudrucken, die anzeigt, wie der geschuldete Betrag mit der Zeit abnimmt, vorausgesetzt, daß die Zahlungen rechtzeitig erfolgen.
Zwischen den einzelnen Zahlungen berechnen Sie den täglichen Zins in folgender Weise:

```
70 FOR I=1 TO 365/N
80 B=B+B*D
90 NEXT
```

Geht eine Zahlung ein, ziehen Sie diese vom geschuldeten Betrag ab:

```
100 B=B-P
110 K=K+1
120 PRINT K,B
```

Zeile 120 gibt an, um die wievielte Zahlung es sich handelt und welche Restschuld nach der Zahlung verbleibt. Als nächstes überprüfen Sie, ob die Restschuld noch den Wert Null übersteigt; wenn das der Fall ist, wird mit dem nächsten Rückzahlungszeitraum fortgefahren.

```
130 IF B>0 THEN GOTO 70
140 PRINT "SCHULD IST ABBEZAHLT"
```

Geben Sie das Programm ein, und lassen Sie es laufen.

```
]RUN
WIEVIEL WOLLEN SIE LEIHEN? 1000
DER JAEHRLICHE ZINSSATZ? 15.5
WIE VIELE ZAHLUNGEN PRO JAHR? 12
HOEHE DER ZAHLUNGEN? 145
1            867.818483
2            733.9426
3            598.350632
4            461.020579
5            321.930164
6            181.05682
7            38.3776936
8            -106.130363
SCHULD IST ABBEZAHLT
```

Übungen

1. Ersetzen Sie die Variable B in Zeile 120 durch den Ausdruck INT(B * 1000)/100. Dadurch wird der Input B auf ganze Pfennige abgerundet. Die Funktion INT liefert die größte ganze Zahl, die größer oder gleich der Zahl innerhalb der Klammern ist. Folgendes Beispiel veranschaulicht dies:

   ```
   INT(2.3456*100)/100
   =INT(234.56)/100
   =234/100
   =2.34
   ```

2. Lassen Sie in Zeile 120 noch zwei weitere Angaben ausdrucken, und zwar den Zinsanteil sowie den Tilgungsanteil der jeweiligen Zahlung.
3. Erweitern Sie das Programm aus Übung 2 so, daß es am Ende den Betrag der insgesamt gezahlten Zinsen sowie die Summe aus dem geliehenen Betrag und den gesamten Zinsen ausdruckt.

14. Magische Summen

In diesem Abschnitt üben Sie den Gebrauch von FOR...NEXT-Schleifen und wiederholen nebenbei noch ein bißchen Mathematik. Die gestellten Probleme sind von folgender Art:

```
S= 3 + 5 + 7 + ... + 33
```

Dabei müssen Sie die Maschine dazu bringen, die fehlenden Zahlen, die durch +...+ angedeutet sind, aufzusummieren.

Dazu schreiben Sie ein kurzes Programm:

```
10 FOR N = 3 TO 33 STEP 2
20 S = S + N
30 NEXT N : PRINT "DIE SUMME IST ",S
```

Lassen Sie das Programm mit Hilfe des Kommandos RUN laufen; Sie sollten

```
DIE SUMME IST 288
```

als Ausgabe erhalten.

Hier ein ähnliches Beispiel:

```
S = 10 + 15 + 20 + ... + 55
```

Die Antwort ist diesmal 1495. Lesen Sie nicht weiter, bevor Sie diese Antwort nicht bekommen haben.
Nun betrachten Sie folgende Summe:

$$S = 10^2 + 20^2 + 30^2 + ... + 100^2$$

Der Trick bei der Programmierung dieser Summe besteht darin, den Schleifenindex von 10 über 20, 30 usw. laufen zu lassen, also jedesmal um 10 zu erhöhen, und das Quadrat des Schleifenindex zu bilden, bevor man ihn zur Summe hinzuaddiert.
Die Lösung sieht also folgendermaßen aus:

```
10 FOR I = 10 TO 100 STEP 10
20 S = S + I^2 : NEXT I : PRINT "DIE
   SUMME IST ";S
```

Der Programmaufbau wird etwas komplizierter, hat man es mit magischen Summen folgender Art zu tun:

$$S = 4^2 + 6^3 + 8^4 + ... + 12^?$$

Beachten Sie, daß die Potenz, in die die einzelnen Zahlen erhoben werden, mit jeder neuen Zahl um Eins steigt. Das Fragezeichen soll andeuten, daß wir nicht unbedingt wissen, in die wievielte Potenz die letzte Zahl erhoben werden muß. Den Schleifenindex lassen Sie wieder von 4 in Zweierschritten bis 12 laufen. Hier eine mögliche Lösung:

```
10 FOR I = 4 TO 12 STEP 2
20 S = S + I^(P+2)
30 P=P+1
40 NEXT I : PRINT "DIE SUMME IST ";S
```

Wie Sie sehen, macht das Programm wieder von dem Umstand Gebrauch, daß bei Beginn eines Programmlaufs sämtliche Variablen auf Null gesetzt werden. So wissen Sie, daß S und P am Anfang den Wert 0 haben.

15. Zeichenreihenvariablen

Außer mit Zahlen kann BASIC auch mit Worten arbeiten, die sich aus den verschiedenen Zeichen auf der Tastatur zusammensetzen. Diese Worte nennt man Strings oder Zeichenreihen, wobei es sich dabei nicht unbedingt um ein verständliches Wort handeln muß; stellen Sie sich vor, daß jedes Zeichen einer Zeichenreihe nur mit dem nächsten Zeichen verbunden ist, also die einzelnen Zeichen einer Zeichenreihe sozusagen eine Kette bilden.
Zeichenreihen werden in Zeichenreihenvariablen abgespeichert. In Applesoft BASIC muß der Variablenname einer Zeichenreihenvariablen auf ein Dollarzeichen ($) enden, wie in diesem Beispiel:

```
50 A$ = "HALLO"
```

Durch diese Zuweisung wird der String HALLO in der Speicherzelle mit der Bezeichnung A$ abgelegt. Die Anführungszeichen (") geben lediglich an, wo der String anfängt und wo er aufhört; sie sind nicht Bestandteil des Strings selbst.
Die Anzahl der Zeichen einer Zeichenreihe markieren deren Länge. Der String HALLO hat die Länge 5. Wenn man einen String manipuliert, ist es oft notwendig, seine Länge zu kennen; in Applesoft BASIC erreicht man dies durch folgende Zuweisung:

```
60 L = LEN(A$)
```

In der Regel setzen Sie den Namen der Zeichenreihenvariablen in Klammern, und die Funktion LEN liefert die Anzahl der Zeichen der Zeichenreihe.
Lassen Sie folgendes Programm laufen:

```
10 REM  DER LEERE STRING
20 N$ = ""
30 PRINT LEN (N$);N$; LEN ("HALLO")
]RUN
05
```

An diesem Beispiel zeigt sich, wie Applesoft BASIC mit Zeichenreihen ohne Zeichen, auch leere Strings genannt, umgeht. Ein leerer String wird dadurch definiert, daß keine Zeichen zwischen den Anführungszeichen stehen. Zu Beginn der Ausführung eines Programms werden von der Maschine automatisch sämtliche Zeichenreihenvariablen mit dem leeren String besetzt.
Stringmanipulation erfordert zumeist, daß Sie mit irgendwelchen Teilstrings eines Strings arbeiten. Applesoft BASIC kennt drei Kommandos, die es Ihnen ermöglichen, sich den linken, den mittleren oder den rechten Teil eines Strings zu beschaffen, wie das folgende Programm zeigt:

```
10 REM   DEMONSTRATIONSPROGRAMM FUER
         LEFT$, MID$ & RIGHT$
20 A$ = "MISSISSIPPI"
30 L$ = LEFT$ (A$,3)
40 M$ = MID$ (A$,4,4)
50 R$ = RIGHT$ (A$,4)
60 PRINT L$: PRINT M$:, PRINT R$
70 PRINT R$;M$;L$
]RUN
MIS
SISS
IPPI
IPPISISSMIS
```

In Zeile 30 gibt der Wert 3 die Anzahl der Zeichen an, die von links gelesen werden, und in Zeile 50 der Wert 4 die Anzahl der Zeichen, die von rechts genommen werden.

Im Gegensatz dazu repräsentiert in Zeile 40 die erste Zahl 4 die laufende Nummer des Zeichens im String, bei der begonnen werden soll, und die zweite Zahl 4 die Anzahl der Zeichen, die gelesen werden. Geben Sie das Programm ein, und lassen Sie es laufen.
Solche Stringmanipulation benötigt man, wenn man Namen verarbeitet. Es kann beispielsweise nötig sein, Vornamen in einer Personenliste zu entfernen, um diese alphabetisch nach Nachnamen zu sortieren. Betrachten Sie dazu auch Übung 3.

Übungen

1. Geben Sie Ihren ersten und, falls es Ihnen möglich ist, auch Ihren zweiten Vornamen sowie Ihren Nachnamen separat ein. Dann lassen Sie Ihren vollständigen Namen ausdrucken.
2. Geben Sie einen vollständigen Namen ein und speichern Sie ihn in einer einzigen Stringvariablen ab. Bestimmen Sie, aus wie vielen Buchstaben der Vorname besteht, indem Sie die Zeichen zählen, bis Sie auf ein Leerzeichen stoßen. Drucken Sie dann diese Zahl aus. Testen Sie Ihr Programm mit verschiedenen Namen.
3. Geben Sie einen vollständigen Namen wie in Übung 2 ein. Drucken Sie dann den Nachnamen aus. Achten Sie darauf, daß wirklich nur der Nachname ausgegeben wird, auch wenn mehrere Vornamen eingegeben wurden.
4. Lassen Sie Ihr Lieblingslied erfragen. Drucken Sie dann, wie oft der Buchstabe A in der Antwort vorkommt.

16. Eindimensionale Felder

Früher oder später werden Sie es mit einem Satz von Zahlen zu tun bekommen, die beispielsweise verschiedene Messungen ein und derselben Größe darstellen, wie etwa die Außentemperatur vor Ihrem Haus zu verschiedenen Tageszeiten. Haben Sie es mit sehr großen Zahlensätzen dieser Art zu tun, ist es wenig vorteilhaft, jeder dieser Zahlen einen eigenen Variablennamen zuzuordnen, den Sie sich dann merken müssen.
Die Lösung dieses Problems, wie sie die meisten Programmiersprachen erlauben, besteht darin, einen zweiteiligen Namen zu verwenden. Der erste Teil wird als Feldname bezeichnet, er ist für alle Werte des Feldes gleich; der zweite Teil des Namens heißt Indexteil, hierbei muß es sich um eine nichtnegative, ganze Zahl handeln.
Die einzelnen Zahlen eines eindimensionalen Feldes kann man sich als eine Aufreihung gleichartiger Speicherzellen vorstellen, sozusagen ein Hotel für Zahlen. Mit dem Feldnamen bezieht man sich auf die Gesamtheit der Speicherzellen (das entspricht dem Hotelnamen), die einzelnen Zellen spricht man dann einfach mit Hilfe von Zahlen an, also 0, 1, 2, 3 usw.

A	0	1	2	3	4	5	6	7	8

In allen Versionen von BASIC werden Feldnamen und Indizes in folgender Weise geschrieben:

```
56 X = A(34)
```

Diese Anweisung nimmt die Gleitpunktzahl aus Zelle 34 des Feldes A und überschreibt damit den bisherigen

Inhalt von X. Als Index kann man sowohl Variable als auch Ausdrücke verwenden. Die Anweisung

```
45 BAL = EMPFAENGER(KUNDE)
```

ist in Applesoft BASIC zulässig, jedoch nicht in den meisten anderen Versionen von BASIC. Denken Sie daran, daß die Maschine nur die ersten zwei Zeichen eines Namens registriert, und achten Sie deshalb auf deren Eindeutigkeit.

Angenommen, Sie haben 30 Zahlen, die Sie in der Maschine unterbringen wollen und die alle vom selben Typ sind, beispielsweise die täglichen Temperaturmessungen eines ganzen Monats. Man kann diese Zahlen natürlich mit T_1, T_2, T_3 bis hin zu T_{30} bezeichnen, schon ehe man sie in die Maschine eingibt, beispielsweise, um die Werte ohne Schwierigkeiten mit anderen vergleichen zu können. In der Mathematik hat es sich durchgesetzt, die Zählnummer gegenüber dem Feldnamen tiefer gesetzt zu schreiben, so daß der Schriftsetzer keine Klammern benötigt.

Die folgende Routine dient zur Eingabe von 30 Zahlen in ein Feld, das mit dem Feldnamen T bezeichnet wird:

```
10 REM  EINGABE VON 30 ZAHLEN IN FELD T
20 DIM T(30)
30 PRINT "GEBEN SIE HINTER DEM
   FRAGEZEICHEN JEWEILS EINE ZAHL EIN"
40 FOR I = 1 TO 30
50 INPUT T(I)
60 NEXT I
70
```

Übungen

1. Fragen Sie den Benutzer nach 12 Zahlen. Legen Sie diese in einem Feld namens N ab, wobei Sie bei Speicherplatz 1 beginnen. Vergessen Sie nicht, am Anfang Ihres Programms eine Dimensionsanweisung zu schreiben. Sind alle Zahlen eingegeben, soll der Inhalt von N ausgedruckt werden, und zwar jeweils ein Wert pro Zeile.
2. Lassen Sie Ihr Programm die größte Zahl in dem Feld N aus Übung 1 finden. Ein Weg, dies zu erreichen, besteht darin, N (1) in eine Hilfsvariable T zu speichern und dann eine FOR...NEXT-Schleife dazu zu verwenden, die übrigen Speicherplätze von 2 bis 12 durchzuschauen. Findet man ein größeres Element als das, das in T steht, weist man es der Variablen T zu. Wenn die Schleife abgearbeitet ist, wird T das größte Element in Feld N enthalten.
3. Erweitern Sie das Programm aus Übung 2 so, daß auch der Durchschnitt aller 12 Werte ermittelt wird.
4. Bauen Sie ein eindimensionales Feld von Strings auf, indem Sie den Benutzer nach seinen sieben Lieblingsfilmstars fragen. Schreiben Sie dann ein Programmstück, das folgendes leistet:
   a) Das Ausdrucken aller sieben Namen.
   b) Das Ausdrucken des längsten Namens.
   c) Das Ausdrucken aller Namen, die mit dem Buchstaben A beginnen.
   d) Das Ausdrucken der Angabe, wie oft der Buchstabe E in sämtlichen Namen insgesamt vorkommt (nur für Fortgeschrittene).

17. Debitorenbuchhaltung

Wenn Sie im Geschäftsleben stehen und Geld von verschiedenen Seiten zu bekommen haben, dann nennt man diese Geldquellen auch Ihre Debitoren. Für jeden der Debitoren führen Sie in der Regel ein eigenes Konto, das den jeweils gerade ausstehenden Betrag anzeigt und das Sie stets auf neuestem Stand halten, indem Sie die dieses Konto betreffenden Rechnungen und Zahlungen registrieren.
Wollen Sie, daß Ihr Apple Konten dieser Art führt, müssen Sie, wenn eine Zahlung auf ein Konto eingeht, die Kontonummer und den Betrag eingeben. Das Programm ändert den Kontostand, indem es den Betrag der Zahlung vom Kontostand abzieht. Sie nehmen also eine Gutschrift auf dem Konto vor.
Verarbeiten Sie dagegen eine Rechnung, müssen Sie natürlich ebenfalls Kontonummer und auszuzahlenden Betrag eingeben. Das Programm wird dann den Betrag der Rechnung zum Kontostand hinzuaddieren. Sie haben das Konto in diesem Fall belastet. Im anschließenden Beispiel sind zehn Konten und 100 Kontenbewegungen vorgesehen. Das Programm weist folgende Anfangszeilen auf:

```
10 REM    ***DEBITORENBUCHHALTUNG***
20 DIM B(10) :REM   KONTOSTAND
30 DIM T(100)  :REM   BEWEGUNGSART
40 DIM C(100)  :REM   BEWEGUNGSKONTO
50 DIM A(100)  :REM   BETRAG
```

Beachten Sie, daß das Programm die Daten jeder einzelnen Bewegung aufhebt, wozu es drei eindimensionale Felder benutzt.

Aus Testzwecken nehmen Sie die folgenden Anfangsdaten für die Kontenstände an:

```
60 DATA 245,873,54,4442,1733
70 DATA  -76,0,531,1208,320
80 FOR  I=1 TO 10:READ B(I):NEXT
```

Nach Ausführung von Zeile 80 sind die Kontenstände mit den obigen Werten vorbesetzt. Sie sind jetzt in der Lage, die Bewegungen zu verarbeiten.

```
90 K=1 :REM  K IST DIE NUMMER DER
   BEWEGUNG
100 INPUT "BEWEGUNGSART";T(K)
110 IF T(K)=0 THEN GOTO 300
120 INPUT "KONTONUMMER (1 BIS 10)";C(K)
130 INPUT "BETRAG:";A(K)
```

Jetzt ist die Bewegung auf dem Konto gespeichert. Als nächstes ändern Sie den Kontenstand

```
140 IF T(K) = 1 THEN B(C)=B(C)+A(K)
150 IF T(K) = 2 THEN B(C)=B(C)-A(K)
```

und machen dann mit den nächsten Kontenbewegungen weiter:

```
160 K=K+1 : GOTO 100
300 M = K - 1
```

Sind alle Kontenbewegungen eingegeben und verarbeitet, stehen Sie vor der Anweisung mit Zeilennummer 300. Der Wert von M bezeichnet die Anzahl der Bewegungen. Diese Anzahl hat den Wert K-1, weil der K-te Vorgang keine echte Kontobewegung darstellt. Das Programm besteht aus einer Liste der Bewegungen zu jedem einzelnen Konto; studieren Sie es sorgfältig:

```
310 FOR I=1 TO 10
315 PRINT "BEWEGUNGEN AUF KONTO ";I
320 FOR K=1 TO M
330 IF C(K)=I THEN PRINT C(K),A(K)
340 NEXT
350 NEXT
```

```
]LIST

10 REM    ***DEBITORENBUCHHALTUNG***
20 DIM B(10): REM   KONTOSTAND
30 DIM T(100): REM   BEWEGUNGSART
40 DIM C(100): REM   BEWEGUNGSKONTO
50 DIM A(100): REM   BETRAG
60 DATA  245,873,54,4442,1733
70 DATA -76,0,531,1208,320
80 FOR I = 1 TO 10: READ B(I): NEXT
90 K = 1: REM   K IST DIE NUMMER DER
   BEWEGUNG
100 INPUT "BEWEGUNGSART:";T(K)
110 IF T(K) = 0 THEN GOTO 300
120 INPUT "KONTONUMMER (1 BIS 10):";C(K)
130 INPUT "BETRAG:";A(K)
140 IF T(K) = 1 THEN B(C) = B(C) + A(K)
150 IF T(K) = 2 THEN B(C) = B(C) - A(K)
160 K = K + 1: GOTO 100
300 M = K - 1
310 FOR I = 1 TO 10
315 PRINT "BEWEGUNGEN AUF KONTO ";I
320 FOR K = 1 TO M
330 IF C(K) = I THEN  PRINT C(K),A(K)
340 NEXT
350 NEXT
```

```
]RUN
BEWEGUNGSART:1
KONTONUMMER (1 BIS 10):1
BETRAG:25
BEWEGUNGSART:1
KONTONUMMER (1 BIS 10):1
BETRAG:456
BEWEGUNGSART:2
KONTONUMMER (1 BIS 10):6
BETRAG:322
BEWEGUNGSART:2
KONTONUMMER (1 BIS 10):6
BETRAG:763
BEWEGUNGSART:1
KONTONUMMER (1 BIS 10):3
BETRAG:456
BEWEGUNGSART:1
KONTONUMMER (1 BIS 10):7
BETRAG:6789
BEWEGUNGSART:2
KONTONUMMER (1 BIS 10):5
BETRAG:653
BEWEGUNGSART:2
KONTONUMMER (1 BIS 10):7
BETRAG:23
BEWEGUNGSART:0
KBEWEGUNGEN AUF KONTO:1
1                 25
1                 456
BEWEGUNGEN AUF KONTO:2
BEWEGUNGEN AUF KONTO:3
3                 456
BEWEGUNGEN AUF KONTO:4
BEWEGUNGEN AUF KONTO:5
5                653
```

```
BEWEGUNGEN AUF KONTO:6
6           322
6           763
BEWEGUNGEN AUF KONTO:7
7           6789
7           23
BEWEGUNGEN AUF KONTO:8
BEWEGUNGEN AUF KONTO:9
BEWEGUNGEN AUF KONTO:10
```

Übungen

1. Ändern Sie obiges Programm so ab, daß es den derzeitigen Kontostand zu jedem Konto ausdruckt, ehe es die Bewegungen auf jedem Konto ausgibt. Sie müssen dazu nur Zeile 315 ändern.
2. Ändern Sie Zeile 330 des obigen Programms so ab, daß anstelle der Kontonummer C (K) die Bewegungsart T (K) gedruckt wird.
3. Berücksichtigen Sie auch das Datum der eingehenden Zahlungen beziehungsweise Rechnungen. Fügen Sie dazu ein weiteres Feld und eine weitere INPUT-Anweisung in das Programm ein. Geben Sie das Datum bei der Ausgabe mit aus. Verwenden Sie dabei das Format MMTT (für Monat und Tag), um die Ausgaben später nach dem Datum sortieren zu können. Dabei ist das Kommando TAB (N) von Vorteil, um auf Spalte N springen zu können. Eine Möglichkeit besteht darin, nur ein Datenelement pro PRINT-Anweisung zu drucken, vor dieses ein TAB-Kommando zu setzen und dahinter einen Strichpunkt. Beim letzten zu druckenden Datenelement einer Zeile lassen Sie den Strichpunkt weg. Auf diese Weise können Sie die Ausgabe leicht abändern. Spätestens dann, wenn Sie in einer Zeile mehr als fünf Daten ausgeben, werden Sie Überschriften für die Spalten benötigen. Auch das kann man mit PRINT-Anweisungen erledigen.

18. Hauptspeichersortieren

Nachdem Sie nun eine gewisse Erfahrung mit eindimensionalen Feldern gewonnen haben, ist es an der Zeit, ein verbreitetes Sortierverfahren kennenzulernen, das sogenannte Hauptspeichersortieren. Es gibt dabei mehrere Varianten ein und desselben Grundprinzips, das darin besteht, Paare von Werten aus dem Feld zu vergleichen und die Werte, wenn sie in falscher Reihenfolge stehen, zu vertauschen. Wenn das für ein Paar von Werten geschehen ist, fährt das Programm mit einem anderen Wertpaar fort. Schließlich wird das Feld auf diese Weise sortiert sein.

Zunächst bringen Sie die Zahlen in Feld A:

```
10 REM  HAUPTSPEICHERSORTIEREN
20 DIM A(20)
30 INPUT "WIE VIELE ZAHLEN? ";N
40 IF N>20 THEN PRINT "ZU VIELE";GOTO 30
50 FOR I=1 TO N
60 INPUT "BITTE ZAHL EINGEBEN: ";A(I)
70 NEXT
```

An dieser Stelle enthält N also die Anzahl der zu sortierenden Werte, und die unsortierten Werte stehen in den Zellen A (1), A (2), ... A (N).

Das Verfahren, das Sie in diesem Beispiel anwenden, bestimmt den größten Wert im Feld A und stellt ihn an die erste Stelle, indem die erste Position mit allen anderen der Reihe nach (2, ..., N) verglichen wird. Ist der erste Wert kleiner als der Wert, mit dem er gerade verglichen wird, werden beide miteinander vertauscht. Wenn man an das Ende des Felds gekommen ist, enthält die Position Eins den größten Wert der gespeicherten Zahlen.

Als nächstes vergleichen Sie die zweite Position und füh-

ren auch hier wieder die notwendigen Vertauschungen durch.
Auf diese Weise fahren Sie fort, bis Sie die Position N-1 mit allen folgenden vergleichen müssen. Dabei ist offensichtlich nur noch ein einziger Vergleich nötig.
Überprüfen Sie das folgende Programm, und stellen Sie sicher, daß es auch tatsächlich in geforderter Weise funktioniert:

```
80 FOR P = 1 TO N-1
90 FOR I = P+1 TO N
100 IF A(P) > = A(I) THEN GOTO 120
110 T = A(I):A(I) = A(P):A(P) = T
120 NEXT
130 NEXT
140 FOR I = 1 TO N: PRINT A(I): NEXT
```

Geben Sie das Programm ein, und lassen Sie es laufen.

Übungen

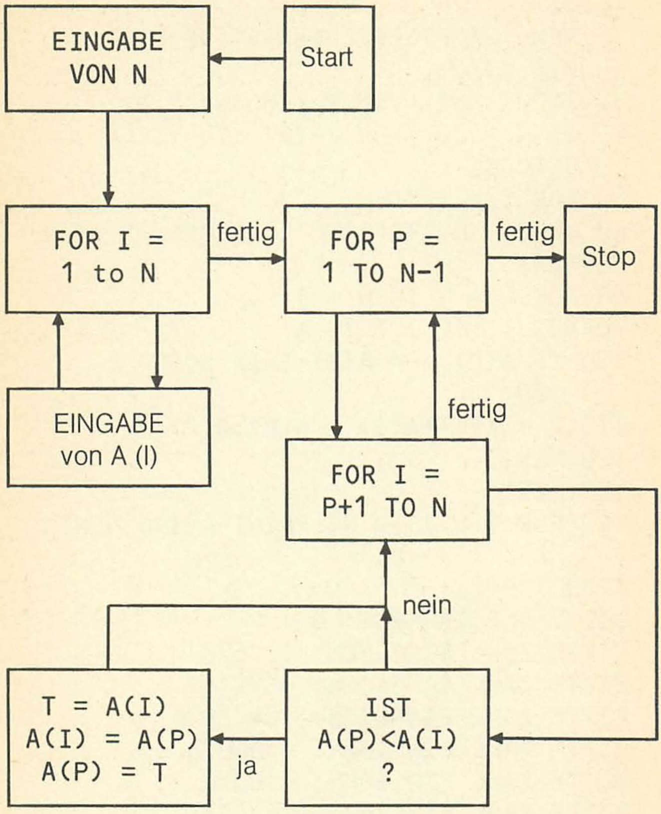
1. Versuchen Sie, aus den Zeilen 80 bis 140 eine möglichst kleine Anzahl von mit Zeilennummern versehenen Anweisungen zu bilden.
2. Ändern Sie das Hauptspeichersortierprogramm so ab, daß schließlich die kleinste Zahl im ersten Feldelement und die größte Zahl im letzten Feldelement steht.
3. Bei einer geringfügig abgeänderten Variante des Blasensortierens läuft man N-1mal durch das Feld A und schaut dabei nur benachbarte Paare von Werten an, wobei man beim ersten Feldelement anfängt. Nach dem ersten Durchlauf steht die kleinste Zahl im letzten Feldelement (also in A (N)). Im zweiten Durchlauf besteht deshalb keine Notwendigkeit, über das Feldelement A (N-1) hinauszugehen. Beim N-1ten Durchlauf schließlich muß man nur noch A (1) und A (2)

miteinander vergleichen. Ändern Sie das Sortierprogramm so ab, daß es nach diesem Verfahren arbeitet.

```
]LIST
10 REM  HAUPTSPEICHERSORTIEREN
20 DIM A(20)
30 INPUT "WIE VIELE ZAHLEN? ";N
40 IF N > 20 THEN PRINT "ZU VIELE";
   GOTO 30
50 FOR I = 1 TO N
60 INPUT "BITTE ZAHL EINGEBEN: ";A(I)
70 NEXT
80 FOR P = 1 TO N - 1
90 FOR I = P + 1 TO N
100 IF A(P) > = A(I) THEN GOTO
    120
110 T = A(I):A(I) = A(P):A(P) = T
120 NEXT
120 NEXT
140 FOR I = 1 TO N: PRINT A(I): NEXT

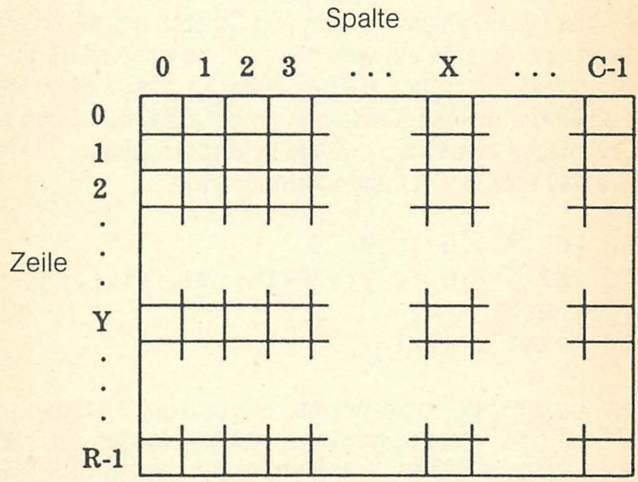
]RUN
WIE VIELE ZAHLEN?: 6
BITTE ZAHL EINGEBEN: 12.678
BITTE ZAHL EINGEBEN: -87.453
BITTE ZAHL EINGEBEN: 104.39
BITTE ZAHL EINGEBEN: -.00076
BITTE ZAHL EINGEBEN: -.0076
BITTE ZAHL EINGEBEN: 56.443
104.39
56.443
12.678
-7.6E-04
-7.6E-03
-87.453
```

Flußdiagramm für das Hauptspeichersortieren:



19. Zweidimensionale Felder

Wie Sie gesehen haben, sind eindimensionale Felder etwas ganz Natürliches, wenn Sie eine Folge von Werten zu einer Größe bearbeiten. Sie können ein eindimensionales Feld entweder als eine Zeile oder als eine Spalte von Zahlen anordnen.
Ein zweidimensionales Feld kann man sich als rechtekkige Anordnung von Zahlen vorstellen, die R Zeilen und C Spalten besitzt:



Der Index Y für die Zeilen läuft von Null bis R-1, der Index X für die Spalten von Null bis C-1. Um beispielsweise den Wert 8.7 in eine bestimmte Zelle, sagen wir in Zeile Y und Spalte X, zu bringen, schreibt man

```
730 GE(X,Y) = 8.7
```

wobei CE der Name des zweidimensionalen Feldes ist.
Als Beispiel bauen wir ein Feld MULT mit 10 Zeilen und 10 Spalten auf, wobei der Wert in Zeile Y und Spalte X das Produkt aus X und Y sein soll.

```
10 REM   MULTIPLIKATIONSTABELLE
20 DIM MULT(9,9)
30 FOR Y = 0 TO 9 :  FOR X = 0 TO 9
40 MULT(X,Y) = X*Y
50 NEXT : NEXT
```

Beachten Sie, daß in der Dimensionsanweisung die tatsächliche Anzahl von Zeilen und Spalten um den Wert 1 vermindert erscheint, weil die Zählung mit Null beginnt. Beachten Sie ferner, daß in Zeile 50 die Angabe des Schleifenindexes unmittelbar hinter NEXT nicht nötig ist. Um dieses zweidimensionale Feld auszugeben, setzen Sie das Programm folgendermaßen fort:

```
60 FOR Y = 0 TO 9
70 FOR X = 0 TO 9 : PRINT MULT(X,Y);"  ";
   : NEXT
80 PRINT : NEXT
```

Geben Sie das Programm jetzt ein, und lassen Sie es mit Hilfe von RUN laufen. Man kann nahezu denselben Effekt mit einem einzigen Kommando, nämlich MAT PRINT MULT, erreichen, aber Sie erhalten damit nicht jene vorteilhafte Anordnung auf dem Bildschirm wie im obigen Beispiel.
Ein guter Beleg für die Nützlichkeit zweidimensionaler Felder ist die Analyse von Häufigkeitsverteilungen. Stellen Sie sich vor, Sie haben einen Text vorliegen, der aus 8 Fragen besteht, zu denen es jeweils 5 mögliche Antworten gibt. Nehmen Sie weiter an, daß sich 100 Studenten

diesem Test unterzogen haben und daß deren Antworten in einem Block von 100 DATA-Anweisungen verschlüsselt sind, wobei jede DATA-Anweisung die 8 Antworten eines bestimmten Studenten enthält.

Wenn Sie den Test nun analysieren, werden Sie mit Sicherheit wissen wollen, wie viele Studenten jede einzelne Frage in welcher Weise beantwortet haben. Wie viele Studenten haben beispielsweise bei Frage Nummer 7 die Antwortmöglichkeit Nummer 3 gewählt? Was war die häufigste Antwort auf die Frage Nummer 2?

Was Sie brauchen, ist ein zweidimensionales Feld H mit 5 Zeilen (je eine Zeile für jede mögliche Antwort) und 8 Spalten. Wenn sämtliche DATA-Anweisungen gelesen worden sind, enthält H (X, Y) die Anzahl der Studenten, die auf die Frage X die Antwort Y gegeben haben.

Wie baut man das Feld H nun auf? Eine Möglichkeit besteht darin, jeweils eine DATA-Anweisung zu lesen und festzustellen, wie der Student die erste Frage beantwortet hat. Hat er beispielsweise die Antwortmöglichkeit 3 gewählt, wird die Zelle, die der Antwortmöglichkeit 3 auf die Frage 1 entspricht, um 1 erhöht, also:

```
H(1,3) = H(1,3) + 1
```

Was auch immer die Antwort auf die erste Frage gewesen sein mag, Sie wählen in jedem Fall die entsprechende Zelle aus und erhöhen ihren Wert um Eins.

Anschließend behandeln Sie die restlichen sieben Fragen in ähnlicher Weise. Wenn sämtliche DATA-Anweisungen gelesen worden sind, dann enthält H die gewünschte Häufigkeitsverteilung. Versuchen Sie ein Programm zu schreiben, das H berechnet und dann ausdruckt. Verwenden Sie ruhig die Dimension H (8, 5) und ignorieren Sie die zusätzliche Zeile und Spalte mit dem Index Null. Vergleichen Sie Ihr Programm mit der Lösung auf den nächsten Seiten:

```
10 REM  HAEUFIGKEITEN DER TESTANTWORTEN
20 DIM H(8,5)
30 FOR S = 1 TO 100   GOSUB 100 : NEXT
40 GOSUB 200  REM  DRUCKEN DES FELDES H
100 REM  VERARBEITUNG DER DATEN
    EINES STUDENTEN
110 FOR C = 1 TO 8 :  READ A
120 H(C,A) = H(C,A) + 1 : NEXT : RETURN
200 REM   DRUCKROUTINE
210 FOR R = 1 TO 5 : FOR C = 1 TO 8
220 PRINT H(C,R); "   "; : NEXT : PRINT :
    NEXT
```

```
]PRINT CHR$(9) "80N"

]LIST

10  REM    ZWEIDIMENSIONALES FELD
20  DIM H(6,5)A(10,6)
30  FOR I = 1 TO 10: REM     FUER JEDEN
    DER ZEHN STUDENTEN
40  REM             DATEIEINTRAEGE
60  READ A(I,1),A(I,2),A(I,3),A(I,4),
    A(I,5),A(I,6)
70  REM             BEARBEITUNG DER
    DATEN DIESES STUDENTEN
80  FOR Q = 1 TO 6:H(Q,A(I,Q)) =
    H(Q,A(I,Q)) + 1: NEXT Q
90  NEXT I: REM   DIE DATEN DIESES
    STUDENTEN SIND BEARBEITET
100  REM   AUSGABE DER HAEUFIGKEIT DER
     TESTANTWORTEN FELD H
110  FOR A = 1 TO 5: FOR Q = 1 TO 6: PRINT
       H(Q,A);"   ";: NEXT Q
```

```
120  PRINT : NEXT A: PRINT "BITTE SEHR"
125  REM
130  DATA  1,2,3,2,5,5
140  DATA  2,2,3,2,5,3
150  DATA  1,2,4,5,1,3
160  DATA  1,4,3,5,5,3
170  DATA  2,2,3,5,1,4
180  DATA  5,2,1,5,4,4
190  DATA  1,2,3,5,4,3
200  DATA  1,2,4,2,2,2
210  DATA  2,1,3,5,1,3
220  DATA  2,2,2,2,2,2

]
5  1  1  0  3  0
4  8  1  4  2  2
0  0  6  0  0  5
0  1  2  0  2  2
1  0  0  6  3  1
BITTE SEHR
```

20. Schwachauflösende Farbgraphik

Nachdem Sie nun zweidimensionale Felder kennengelernt haben, wird Ihnen die Benutzung der Applesoft Farbgraphik keine Schwierigkeiten bereiten. Der Bildschirm besitzt bei schwacher Auflösung 40 Zeilen (0 bis 39) und 40 Spalten (0 bis 39). Der »Wert« in der Zelle (X, Y) kann einen von 16 Farben (auf die wir später eingehen werden) annehmen. Man nennt einen solchen Wert Bildelement oder Pixel (Kurzform für das englische »picture element«).
Lassen Sie für den Anfang das folgende Programm mit einem Farbfernsehbildschirm laufen. Sie müßten dann alle 16 Farben als senkrechte Streifen von je zwei Pixel Breite sehen (Schwarz heißt in diesem Fall, daß der Bildschirm aussieht wie im ausgeschalteten Zustand).

```
10 REM  STREIFENERZEUGUNG AUF FARBFERN-
   SEHER
20 GR : REM  EINSCHALTEN SCHWACHAUFLOE-
   SENDE GRAPHIK
30 FOR C = 0 TO 15 : REM   FUER ALLE
   FARBEN
40 COLOR = C    REM : FARBEINSTELLUNG
50 FOR Y = 0 TO 39 : REM   ZEICHNEN DES
   STREIFENS
60 PLOT 2*C,Y  :  PLOT 2*C+1,Y  :  REM
   AUSGABE EINES PIXELPAARES
70 NEXT :  REM   ENDE DES STREIFENS
80 NEXT :  REM   ENDE ALLER 16 FARBEN
```

Im folgenden sind einige der zur Verfügung stehenden Kommandos aufgeführt.

Kommandos für schwachauflösende Graphik

COLOR = x

Hierbei ist x eine Zahl zwischen 0 und 15 mit folgender Bedeutung:

0 schwarz
1 magentarot
2 dunkelblau
3 purpurrot

4 dunkelgrün
5 grau
6 mittelblau
7 hellblau

8 braun
9 orange
10 grau
11 rosa

12 grün
13 gelb
14 aquamarin
15 weiß

Die einmal gewählte Farbe wird so lange verwendet, bis Sie sie durch ein anderes COLOR-Kommando ändern.

GR	Das Kommando schaltet die schwachauflösende Graphik ein, löscht den Bildschirm und setzt den Kursor in das Textfenster. Der Farbwert wird automatisch auf Schwarz (0) gesetzt.
HTAB x	Das Kommando setzt den Kursor in Spalte x (x = 0, 1, 2, ..., 39).
PLOT x, y	Das Kommando setzt ein Pixel mit der gerade ein gestellten Farbe auf Position x, y.
VLIN Y_1, Y_2, AT X	Das Kommando zieht eine senkrechte Linie von (X, Y_1) nach (X, Y_2).
HLIN X_1, X_2 AT Y	Das Kommando zieht eine waagrechte Linie von (X_1, Y) nach (X_2, Y).
VTAB y	Das Kommando setzt den Kursor in Zeile Y (Y = 0, 1, 2, ..., 39).

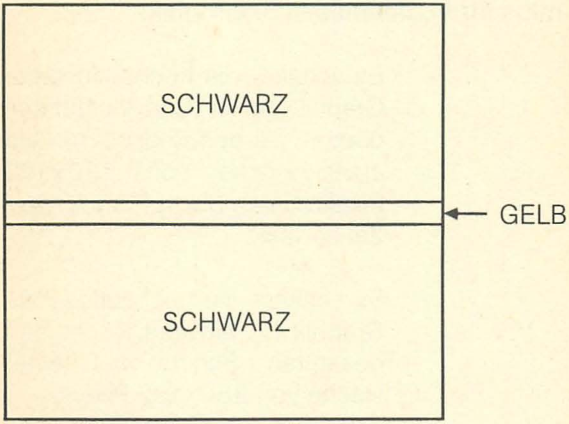
Wie man waagrechte Linien zieht

Diese Arbeit erledigt Applesoft BASIC automatisch, aber es ist lehrreich zu sehen, wie man es selbst machen müßte. Dazu benötigt man die Zahlen der X-Koordinaten ganz links und rechts sowie die der Y-Koordinate der Linie. Nehmen Sie an, die entsprechenden Variablen heißen HL, HR und V. Bedenken Sie, daß Applesoft BASIC nur die ersten beiden Zeichen von Variablennamen berück-

sichtigt, und daß das zweite Zeichen eine sehr nützliche mnemotechnische Hilfe sein kann.
Angenommen, die schwachauflösende Applesoftgraphik (40x40) sei eingeschaltet, und Sie wollen eine gelbe Linie über den ganzen Schirm zeichnen. Sie erreichen dies mit folgendem Programmstück:

```
10 REM    WAAGRECHTE GELBE LINIE
20 GR
30 COLOR = 13
40 FOR H = 0 TO 39 : PLOT H,9  :  NEXT
50 PRINT "BITTE SEHR"
```

Lassen Sie das Programm laufen; Sie sollten folgende Ausgabe erhalten:



BITTE SEHR

21. Hochauflösende Farbgraphik

Wie Sie gesehen haben, kann man mit schwachauflösender Graphik zwar eine große Palette an Farben erzeugen, aber die Auflösung des Bildschirms reicht nicht für anspruchsvolle Graphik aus. Zu diesem Zweck können Sie bei Applesoft BASIC hochauflösende Graphik generieren, wozu Ihnen 280 mal 160 Pixel zur Verfügung stehen, die eine von sechs verschiedenen Farben annehmen können, nämlich Schwarz, Weiß, Orange, Rosa, Blau und Grün.

Kommandos für hochauflösende Graphik:

HGR	Es schaltet die hochauflösende Graphik (1) ein und löscht den oberen Teil des Bildschirms auf einer Fläche von 280x160 Pixels. Unten bleibt Platz für vier Zeilen Text.
HGR2	Es schaltet die hochauflösende Graphik (2) ein und löscht den gesamten Schirm auf einer Fläche von 280x192 Pixels.
HCOLOR = x	Dient dem Einstellen der Farbe (zulässige Werte sind 0 bis 7).
HPLOT x, y	Setzt einen farbigen Punkt auf die Position mit der waagrechten Koordinate x und der senk-

	rechten Koordinate y. 0,0 ist dabei die linke obere Ecke.
HPLOT X_1, Y_1	Ziehen einer Linie vom Punkt X_1, y1 zum Punkt X_1, Y_2 in der gewählten Farbe.
HPLOT x, y	Setzt einen farbigen Punkt auf die Position x, y.
HPLOT TO x, y	Zieht eine Linie vom zuletzt gezeichneten Punkt nach x, y in der gewählten Farbe.

Zur Veranschaulichung für ein Programm mit hochauflösender Graphik kann das untenstehende Beispiel dienen, mit dessen Hilfe Sie beliebige Linien auf dem hochauflösenden Bildschirm 2 ziehen können.
Die Programmierung ist sehr einfach. Setzen Sie zunächst einen Punkt in die linke obere Ecke:

```
10 REM   ***BELIEBIGE LINIEN***
20 HGR 2:HCOLOR=7
30 HPLOT 0,0
```

Wenn Sie wollen, können Sie diese Anweisung sofort ausführen lassen. Sie müßten einen kleinen weißen Punkt in der linken oberen Ecke Ihres Bildschirms sehen.
Als nächstes beschaffen Sie sich eine beliebige waagrechte Koordinate x und eine beliebige senkrechte Koordinate y mit Hilfe folgender Anweisungen:

```
30 X=INT(279*RND(1))
40 Y=INT(189*RND(1))
```

RND (1) erzeugt eine beliebige Zahl zwischen 0 und 1, und die Funktion INT schneidet die Stellen hinter dem Komma ab.
Nun müssen Sie nur noch eine Linie von dem zuletzt gezeichneten Punkt zu den Koordinaten x, y ziehen:

```
50 HPOLT TO X,Y
60 GOTO 30
```

Lassen Sie das Programm laufen. Es macht Spaß, zuzuschauen. In diesem Fall muß die Maschine schon eine ganze Menge Arbeit ausführen, wohingegen Ihre Anweisungen sehr einfach sind; aber genau das erwartet man ja von einer guten Maschine.

Übungen

1. Ändern Sie das obige Programm so ab, daß es die Linien in einer zufällig gewählten Farbe zeichnet.
   Hinweis: Setzen Sie die Anweisung HCOLOR = INT (7 * RND (1)) an die richtige Stelle.
2. Verwenden Sie dieselbe Technik, um auf dem hochauflösenden Bildschirm 2 zufällige Quadrate zu zeichnen. Jedes Quadrat sollte eine Seitenlänge von 50 Pixels haben. Es macht nichts, wenn die Quadrate am Rand des Bildschirms nicht voll sichtbar sind; der verschwindende Teil taucht in diesem Fall auf der gegenüberliegenden Seite des Bildschirms auf.

22. Systemkommandos

Nachdem Sie nun einige Programme studiert und ausprobiert haben, ist es an der Zeit, von einigen Dienstleistungen des Computers Gebrauch zu machen, die das Leben des Benutzers erheblich erleichtern. Es gibt zwei Arten von Kommandos: Systemkommandos, die auf einem fertiggestellten Programmtext operieren, und Editierkommandos, die auf Einzelzeichen des sich entwikkelnden Programms operieren.
Nachfolgend finden Sie eine Liste der wichtigsten Systemkommandos, die Sie für den Anfang benötigen. Interessieren Sie die weiteren Möglichkeiten, die Applesoft BASIC bietet, studieren Sie die Handbücher, die Ihrem Apple II und der auf Applesoft BASIC laufenden Software beigelegt sind.

Einige Systemkommandos

RUN	Ausführung des Programms, wobei mit der niedrigsten Zeilennummer begonnen wird.
RUN x	Ausführung des Programms, wobei mit der Zeile x begonnen wird.
NEW	Löschen des aktuellen Programms.
LIST	Auflisten des aktuellen Programms.

LIST x-y	Auflisten des aktuellen Programms von Zeile x bis Zeile y.
SAVE	Sichern eines Programms auf Magnetband. Schalten Sie den Kassettenrekorder ein und geben Sie dann SAVE ein. Ein Piepton aus dem Lautsprecher Ihres Apple zeigt den Beginn und das Ende des Sicherungsvorgangs an.
LOAD	Laden eines gesicherten Programms auf Magnetband in den Hauptspeicher. Spulen Sie das Magnetband auf den Anfang des Programms zurück. Sie hören einen Piepton, wenn die Laderoutine auf den Anfang des Programms sowie auf das Ende des Programms stößt.

23. Diskettenbetrieb

Auf Dauer können Sie Ihren Computer kaum ohne Diskettenlaufwerk betreiben. Der Grund liegt vor allem darin, daß der Hauptspeicher des Computers die in ihm abgelegte Information in dem Augenblick verliert, in dem Sie die Maschine ausschalten; sie benötigen daher ein externes Speichermedium, das Ihre Programme und die damit erfaßten Datenmengen, auch als »Dateien« bezeichnet, schnell und dauerhaft zu sichern vermag. Eine Datei können Sie zu jedem denkbaren Zweck anlegen, beispielsweise, um die Telefonnummern von Freunden und Ihre monatlichen Ausgaben zu speichern oder um die Titel Ihrer Privatbibliothek und die Adressen Ihrer Geschäftskunden alphabetisch zu ordnen. Zwar können Sie Programme und Datenmengen auch mit Hilfe eines handelsüblichen Kassettenrekorders auf einer gewöhnlichen Musikkassette speichern und bei Bedarf wieder in den Speicher des Computers laden, jedoch erledigen Sie diesen Vorgang mittels eines Diskettenlaufwerks sehr viel schneller.

Um ein Diskettenlaufwerk betreiben zu können, benötigen Sie eine Systemdiskette; für den Apple II empfiehlt sich das Betriebssystem DOS 3.3, aber auch DOS 3.2 oder jedes äquivalente Betriebssystem ist dafür tauglich. Dieses System ist nötig, damit der Computer das Laufwerk betreiben kann. Auf einer Diskette, die nichts anderes als eine biegsame Plastikscheibe mit magnetisierbarer Oberfläche darstellt, wird Information in bestimmten Abschnitten, Sektoren und Spuren genannt, gespeichert. Es wäre für den Benutzer des Computers sehr beschwerlich, müßte er sich den Speicherplatz der Information auf der Diskette jeweils selbst merken und den Lese/ Schreibkopf des Laufwerks an den entsprechenden Ort

positionieren. Diese Aufgaben erledigt das Betriebssystem bequem und zuverlässig. Es überwacht und ordnet die für den Benutzer nicht sichtbare Dateistruktur auf einer Diskette.
Mittels einer Systemdiskette laden Sie das Betriebssystem in den Speicher des Computers. Schalten Sie Laufwerk und Computer aus, schieben Sie die Systemdiskette in das Laufwerk und schalten Sie den Strom wieder ein; jetzt erfolgt das sogenannte Urladen des Systems. Wenn das Applesoft-Promptzeichen auf dem Bildschirm erscheint, können Sie zum Kennenlernen einige der DOS-Kommandos ausprobieren, die Ihnen durch die Diskette zur Verfügung stehen.
Tippen Sie zunächst das Kommando CATALOG ein und drücken Sie die RETURN-Taste. Dadurch erhalten Sie das Verzeichnis der Dateien, die sich auf der Diskette befinden, beispielsweise in folgender Form:
Neben der Ausgabe des Dateityps erhalten Sie eine Angabe darüber, wieviel Platz (in Sektoren gerechnet) die Datei einnimmt. Ein Sektor ist der sechsunddreißigste Teil des Kreises, den jede der 35 konzentrischen Spuren auf der lesbaren und beschreibbaren Oberfläche der Diskette bildet. Ein solcher Sektor nimmt 256 Bytes bzw. Zeichen auf.
Wenn auf der Diskette im Laufwerk ein Applesoft BASIC-Programm oder ein Integer BASIC-Programm steht (Dateityp A oder I), dann tippen Sie LOAD ein, gefolgt vom Namen der Datei. Das Laufwerk wird daraufhin in Gang gesetzt, und in kürzester Zeit steht Ihnen eine Kopie des Programms im Hauptspeicher zur Verfügung. Sie können das Programm mittels LIST auflisten, sobald das Promptzeichen erscheint.
Tippen Sie RUN, gefolgt vom Dateinamen, ein. So wird das Programm geladen und unmittelbar anschließend ausgeführt. Wollen Sie ein Programm auf Diskette sichern, so müssen Sie SAVE, gefolgt vom Dateinamen,

eingeben und die RETURN-Taste drücken. Erscheint das Promptzeichen wieder auf dem Bildschirm, ist Ihre Datei gesichert. Verwenden Sie dabei den Namen einer Datei, die bereits auf der Diskette gespeichert ist, dann wird das zu sichernde Programm unter diesem Namen auf der Diskette abgelegt und das alte Programm gelöscht. Führen Sie also ein Verzeichnis über die von Ihnen verwendeten Dateinamen.

Auf einer Systemdiskette selbst können Sie keine Programme sichern, da für sie ein permanenter Schreibschutz besteht. Gewöhnliche Disketten weisen am Rand eine Kerbe auf, wodurch dem Laufwerk signalisiert wird, daß Information auf ihr abgespeichert und alte Information überschrieben, also gelöscht werden kann. Überdekken Sie die Kerbe durch einen Klebestreifen, läßt sich die Diskette nicht mehr beschreiben, sie ist damit gegen ein versehentliches Löschen ihrer Information geschützt, wie dies auch bei Systemdisketten der Fall ist. Sofern Sie also Programme sichern wollen, benötigen Sie eine Diskette, die zum einen beschreibbar ist und zum andern bereits formatiert wurde. Darunter versteht man jenen Vorgang, durch den auf der Oberfläche der Diskette jene magnetische Struktur erzeugt wird, die ein geordnetes Abspeichern der Daten ermöglicht.

Um eine neue Diskette zu formatieren, schieben Sie diese einfach in das Laufwerk und schreiben ein kurzes Begrüßungsprogramm, das einige Zeilen ausgibt und das Sie HALLO nennen. Dann geben Sie INIT HALLO ein und drücken die RETURN-Taste; die Diskette wird sich kurz drehen, und der Vorgang ist beendet.

Sie können dies dadurch überprüfen, indem Sie das Kommando CATALOG eingeben. Erhalten Sie nun das Inhaltsverzeichnis der Diskette, so ist diese erfolgreich formatiert.

Auf den folgender Seite finden Sie nochmals die wichtigsten Kommandos in diesem Zusammenhang:

SAVE x Sichern des Programms mit dem Namen x auf der Diskette.

LOAD x Laden des Programms mit dem Namen x von der Diskette.

CATALOG Auflisten des Inhalts der Diskette. Es werden Angaben zu jeder benannten Datei gemacht, die auf der Diskette steht, wobei zu jeder der Dateien die folgenden Angaben in dieser Reihenfolge gemacht werden:
- Schreibschutz (wird durch * angezeigt)
- Dateityp:
- Anzahl der Sektoren
- Name der Datei

Betrachten Sie immer den Katalog, wenn Sie eine neue Diskette einlegen, um sicherzugehen, daß alles in Ordnung ist.

24. Datenspeicherung auf Diskette

Angenommen, Sie verfügen über ein Diskettenlaufwerk und eine DOS-Systemdiskette in der Version 3.2 oder 3.3, so werden Sie rasch Routine im Umgang mit der Speicherung von Daten auf Diskette erlangen. Im nachfolgenden Beispiel speichern Sie eine einfache Zeichenreihe, beispielsweise YPSILON, auf einer Diskette ab, und laden Sie schließlich wieder in den Speicher Ihres Computers. Geben Sie zu diesem Zweck zunächst das Kommando NEW und anschließend folgende Zeilen ein:

```
10 REM DISKETTENSPEICHERUNGSPROGRAMM
20 D$=CHR$(4)
```

Diese so unschuldig aussehende Anweisung besetzt die Stringvariable D$ mit einem speziellen Steuerzeichen. Wenn Sie dieses Steuerzeichen später im Programm benötigen, dann schreiben Sie einfach den Variablennamen D$.

Bei Applesoft BASIC spricht man das Diskettenspeicherungssystem mit Hilfe spezieller PRINT-Anweisungen an, die folgende Gestalt haben:

```
PRINT D$;"MELDUNG"
```

Dabei stellt »meldung« eine noch näher zu behandelnde Zeichenreihe dar.

Nun eröffnen Sie die Datei. Dies geschieht durch das Kommando OPEN, gefolgt vom Dateinamen:

```
30 PRINT D$;"OPEN SESAM"
```

Dieses Kommando sorgt unter anderem dafür, daß Platz für den Datenverkehr zwischen Hauptspeicher und Diskette bereitgestellt wird.
Anschließend müssen Sie der Maschine mitteilen, daß sie die Ausgabe auf Diskette schreiben soll, nicht auf den Drucker oder ein anderes Gerät. Man erreicht das durch eine weitere PRINT-Anweisung:

```
40 PRINT D$; "WRITE SESAM"
```

Jetzt ist es einfach, die Zeichenreihe auf Diskette zu schreiben. Fügen Sie dem Programm die folgende Anweisung hinzu:

```
50 PRINT "YPSILON"
```

Sie schließen die Schreiboperation auf der Diskette durch die Anweisung ab:

```
60 PRINT D$
```

Dabei darf auf die Variable D$ kein Interpunktionszeichen folgen. Schließlich müssen Sie die Datei noch schließen, damit sichergestellt ist, daß alles in Ordnung geht, wenn Sie sie wieder lesen wollen.

```
70 PRINT D$; "CLOSE SESAM"
```

Geben Sie das Programm ein und lassen Sie es laufen. Das Lämpchen am Diskettenlaufwerk wird aufleuchten, die Diskette dreht sich kurz, und schließlich erscheint der Kursor wieder auf dem Bildschirm. Nehmen Sie die Diskette heraus. Sie haben eine Textdatei geschaffen, die einen Datensatz enthält, in dem »YPSILON« steht. Versuchen Sie nun, diese Zeichenreihe wieder in den Hauptspeicher Ihres Computers zu laden.

Tippen Sie NEW ein, um das Programm zu löschen, sowie LIST, um sich zu überzeugen, daß es tatsächlich gelöscht worden ist. Sie können sogar das Gerät ausschalten und anschließend das Betriebssystem neu laden.

Geben Sie nun folgendes Programm ein:

```
10 REM  WIEDERAUFFINDEN VON YPSILON
20 D$ = CHR$ (4)
30 PRINT D$;"OPEN SESAM"
40 PRINT D$;"READ SESAM"
50 INPUT M$; PRINT M$
60 PRINT D$
70 PRINT D$;"CLOSE SESAM"
```

```
]RUN
YPSILON
```

Wenn es Ihnen gelungen ist, YPSILON und das Kursor-Promptzeichen wieder auf den Bildschirm zu bekommen, können Sie sich gratulieren.
Um es zusammenzufassen: Jegliche Steuerung des Diskettenlaufwerks (gleichgültig, um welche Art von Daten es sich handelt) wird mit Hilfe von PRINT D$; »meldung« abgesetzt, wobei D$ das Steuerzeichen CHR$ (4) enthält. Jede Datei, mit der Sie arbeiten, muß einen Dateinamen besitzen, mit OPEN eröffnet werden, ehe sie benutzt wird, und vor Ende des Programms mit CLOSE wieder geschlossen werden. Wenn Sie in die Datei schreiben wollen, dann müssen Sie in einer weiteren PRINT-Anweisung WRITE und den Dateinamen angeben. Daraufhin schickt die Maschine alle in PRINT-Anweisungen ohne D$ vorkommenden Elemente der Reihe nach in die Datei auf der Diskette, genauso, wie wenn man sie über den Drucker ausgibt; das Setzen von Tabulatoren wird in der auf

Diskette gespeicherten Datei allerdings nicht berücksichtigt. Kommata in PRINT-Anweisungen werden in Zusammenhang mit Dateien auf Diskette genauso interpretiert wie Semikolons.
Eine PRINT-Anweisung mit Daten, die nicht durch Komma oder Semikolon abgeschlossen ist, bewirkt, daß ein Vorschubzeichen hinter dem letzten ausgegebenen Datenelement eingefügt wird. Das Vorschubzeichen dient dazu, einer INPUT-Anweisung zu signalisieren, daß alle Datenelemente aufgebraucht sind. Haben Sie mehrere Datenobjekte in einer PRINT-Anweisung versammelt, müssen Sie diese durch Kommas, die Sie als alphanumerische Zeichen angeben, voneinander trennen.
Als Beispiel für ein etwas komplizierteres Programm, das mit einer sequentiellen Datei arbeitet, geben Sie ein Programm an, das die Daten von zehn Personen erfaßt und auf Diskette schreibt.

```
LOAD SCHREIBPROGRAMM
]LIST

10 REM    SPEICHERUNG DER PERSONENDATEN
   AUF DISKETTE
20 D$ = CHR$ (4)
30 PRINT D$;"OPEN DATEI"
40 PRINT D$;"WRITE DATEI"
50 FOR I = 1 TO 10
60 FOR Q = 1 TO 6
70 READ S: PRINT S: NEXT
90 NEXT I: PRINT D$
100 PRINT D$;"CLOSE DATEI"
200 DATA 1,2,3,2,5,5
210 DATA 2,2,3,2,5,3
220 DATA 1,2,4,5,1,3
230 DATA 1,4,3,5,5,3
```

```
240  DATA  2,2,3,5,1,4
250  DATA  5,2,1,5,4,4
260  DATA  1,2,3,5,4,3
270  DATA  1,2,4,2,2,2
280  DATA  2,1,3,5,1,3
290  DATA  2,2,2,2,2,2

]RUN

]LOAD LESEPROGRAMM
]LIST

10  REM    LESEN DER PERSONENDATEN
           VON DISKETTE
20 D$ =CHR$ (4)
30  PRINT D$;"OPEN DATEI"
40  PRINT D$;"READ DATEI"
50  FOR I = 1 TO 10
60  INPUT S1,S2,S3,S4,S5,S6
70  PRINT S1;" ";S2;" ";S3;" ";S4
     ;" ";S5;" ";S6;" "
80  NEXT I

]RUN
1 2 3 2 5 5
2 2 3 2 5 3
1 2 4 5 1 3
1 4 3 5 5 3
2 2 3 5 1 4
5 2 1 5 4 4
1 2 3 5 4 3
1 2 4 2 2 2
2 1 3 5 1 3
2 2 2 2 2 2
```

25. Wie es weitergeht

Neben den in dieser Einführung geschilderten Anwendungen bietet Applesoft BASIC noch zahlreiche weitere Möglichkeiten, wovon Ihnen das Applesoft II BASIC-Handbuch einen ersten Eindruck vermitteln kann. Maßgebend dafür ist, mit welchen Zusatzgeräten und mit welcher Software Sie Ihr System ausstatten.

Für vielfältige Zwecke können ein gutes Textverarbeitungsprogramm und ein Drucker dienen. Der Text dieses Buches wurde unter Verwendung des Epson MX-80-Druckers und des Texteditors Apple Writer geschrieben. Dieses bewährte Programm erlaubt es Ihnen, neue Abschnitte oder Wörter an beliebiger Stelle im Text einzufügen oder Textteile zu verschieben, was, neben anderen Annehmlichkeiten, für eine bequeme Erstellung des Textes von Vorteil ist.

Daneben laufen auf Applesoft BASIC auch verschiedene Spreadsheet-Programme, mit deren Hilfe Sie Finanzplanungen aller Art durchführen können und die zumeist den Grund dafür darstellen, daß Mikrocomputer zunehmend für geschäftliche Zwecke genutzt werden. Sie können sich Ihr Spreadsheet-Programm natürlich auch selbst entwickeln, aber es erfordert einen sehr großen Arbeitsaufwand, bis Programme solcher Art problemlos laufen.

Nahezu unübersehbar ist mittlerweile das Angebot von Computerspielen, aber auch von Graphikprogrammen geworden. Hier bleibt es Ihnen nicht erspart, sich selbst mittels Fachzeitschriften einen Überblick zu verschaffen und mit etwas Glück einen Händler zu finden, der Ihnen das für Ihre Zwecke und Bedürfnisse angemessenste Programm anbietet oder Sie entsprechend berät. Auch wenn mittlerweile zahlreiche Publikationen zu vielen Anwendungsbereichen eines Computers erhältlich sind,

so bleiben Sie doch stets auf das Urteil des Praktikers angewiesen, der Ihnen zuverlässige Angaben darüber machen kann, welches Programm auf Ihrem Betriebssystem läuft und welche Schnittstelle Sie benötigen, um Zusatzgeräte tatsächlich effektiv einsetzen zu können. In welche Richtung und zu welchem Zweck Sie Ihr System ausbauen, hängt letztlich von Ihren Ansprüchen und von der Zeit ab, die Sie für die Beherrschung des Computers und der Software investieren können.

NOTIZEN

NOTIZEN

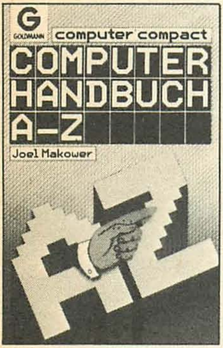
NOTIZEN

NOTIZEN

Goldmann computer compact

Kompetent. Kompakt. Verständlich und verläßlich.

Immer mehr wird der Personal- oder Homecomputer ein nicht mehr wegzudenkendes Hilfsmittel in Haus, Beruf, Schule, Industrie und Verwaltung.
Der Goldmann Verlag vermittelt dem Einsteiger und Fortgeschrittenen in seiner neuen Reihe das Verständnis für die wichtigsten Systeme und Programmiersprachen. Anwendungsbücher, für die jeder Computer-Interessent Verwendung hat!



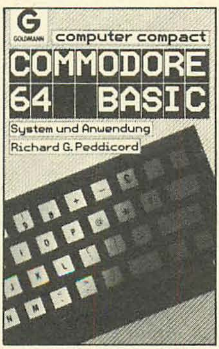
13113



13114



13115



13116

Joseph Reymann
CP / M
Einführung in das populärste Betriebssystem
Best.-Nr. 13117

Steven Manus
IBM PC
Einführung in System und Betrieb
Best.-Nr. 13118

In Vorbereitung:

Richard G. Peddicord
Commodore 64 Graphics
Einführung und Training
Best.-Nr. 13119
(Dezember '84)

Richard G. Peddicord
Apple Basic
System und Anwendung
Best.-Nr. 13120
(Dezember '84)

Richard G. Peddicord
Apple Graphics
Einführung und Training
Best.-Nr. 13121
(Januar '85)

Richard G. Peddicord
Logo
Einführung in die populärste Lernsprache
Best.-Nr. 13122
(Februar '85)

William Urschel
Wordstar
Einführung in das populärste Programm für Textverarbeitung
Best.-Nr. 13123
(März '85)

Carlton Shrum
Supercalc
Einführung und Training
Best.-Nr. 13124
(April '85)

APPLE BASIC

Der Apple II ist besonders erfolgreich wegen seiner Zuverlässigkeit und seinem großen Anwendungsspektrum. Er steht im privaten wie im geschäftlichen Bereich mit an führender Stelle.

Dieses Buch ebnet dem Anfänger den Weg zur optimalen Nutzung der Fähigkeiten des APPLE II. Eine besonders effektive Starthilfe leisten die zahlreichen Übungsbeispiele.

- System und Anwendung
- Programmieren mit BASIC
- Einfache Operationen
- Felder
- Praktische Beispiele

G
GOLDMANN VERLAG

ISB N 3-442-13120-0 DM +009.80

T 3-28-00